COMP.SE.100-EN ItSE

Zoom begins soon...

at 1415 o'clock.

Remote / distant learning 2020.
No contact teaching at ItSE 2020.

Tampereen yliopisto
Tampere University

COMP.SE.100 -EN "ItSE"

Introduction to Software Engineering

2020, 1-2. periods

5 credit units

03-requir-ItSE-2020-v8

Tampereen yliopisto
Tampere University

COMP.SE.100-EN, 2020, course schedule v6c (02.09.2020)

week	lectures	exam	weekly exercises	project assignment (exercise work)	week
35	L1: course basics		--- sign to WE groups ---	sign for project = grouping...	35
36	Project Assignment explained		**WE1**: intro to requirements	grouping, groups to Moodle	36
37	L2: Sw Eng in general		**WE2**: Trello and agile way	group's Trello board ready with product backlog	37
38	L3: requirements		**WE3**: feasibility study and stakeholder analysis	working...	38
39	L4: basic UML diagrams		**WE4**: requirements	working...	39
40	L5: more UML diagrams	EXAM-1	**WE5**: UML diagrams - Use case	working...	40
41	L6: different sw systems	EXAM-1	**WE6**: UML diagrams - concept/entity and navigation	deadline for 1st phase documentation and presentation	41
42	examination week		examination week	examination week	42
43	L7: life cycle models		groups' 1st presentations	groups' 1st phase presentations	43
44	L8: quality and testing	EXAM-2	**WE7**: development processes	feedback group-to-group at PRP, from 1st phase	44
45	L9: project work	EXAM-2	**WE8**: testing and error reporting	deadline for diagrams first versions (Moodle)	45
46	L10: project management		**WE9**: effort estimation	feedback to groups from diagrams (from assistants)	46
47	L11: open source, APIs, IPR		**WE10**: delivery contracts and terms of use	deadline for 2nd phase presentation (PRP)	47
48	L12: embedded systems, IoT	EXAM-3	groups' final presentations	groups' final presentations / feedback g-to-g (PRP)	48
49	L13: recap, summary	EXAM-3	---	final (2.) delivery of project documentation	49
50	examination week		examination week	feedback inside group, student-to-student at PRP	50
51	examination week		examination week	`end of game / game over.`	51
	Lectures: Wed at 1415-16.		Weekly exercises:		
			Mon 0815-10	AUTUMN 2020 (1-2. periods)	
			Mon 1215-14	are remote/distant learning.	
			Tue 0815-10		
			Tue 1415-16		
			Wed 0815-10.		

Tampereen yliopisto
Tampere University

COMP.SE.100-EN (ItSE)
Introduction to Software Engneering

Lecture 3, 16.09.2020

Tensu: remember to start Zoom lecture recording, at 1415

Prefer course Moodle over SISU information.

Students are recommended to follow Moodle News/messages.

Tampereen yliopisto
Tampere University

Course contents (plan)

1. Course basics, intro
2. Sw Eng in general, overview
3. **Requirements**
4. Different software systems
5. Basic UML Diagrams (”Class”, Use Case, Navigation)
6. Life Cycle models
7. UML diagrams, in more detail
8. Quality and Testing
9. Project work
10. Project management
11. Open source, APIs, IPR
12. Embedded systems
13. Recap

Tampereen yliopisto
Tampere University

3. Requirements (SRS = sw req. specification)

- Feasibility Study / Preliminary Analysis
- requirements elicitation
- stakeholders and stakeholder analysis
- Requirements
  - functional
  - non-functional
  - restrictions
- safety vs. security
- usability
- producer’s/developer’s responsibilities
- customer’s responsibilities
- requirements management (e.g. traceability)

Tampereen yliopisto
Tampere University

First, general course matters

Juanita: groups G01-G04

Aleksius: ODD groups; G05,G07,G09,G11,G13,G15,G17,G19,G21,G23,G27

Lauri: EVEN groups; G06,G08,G10,G12,G14,G16,G18,G20,G22,G24,G28

- Trello board and Product Backlog ready at the end of week 37
- invite (not add) your assistant to your Trello board.

WE attendees:

- **Mon 0815-10 9,18,10,**
- **Mon 1215-14 11,12,12,**
- **Tue 0815-10 3, 6, 4,**
- **Tue 1415-16 8,10, 9,**
- **Wed 0815-10 12,11, 9,**

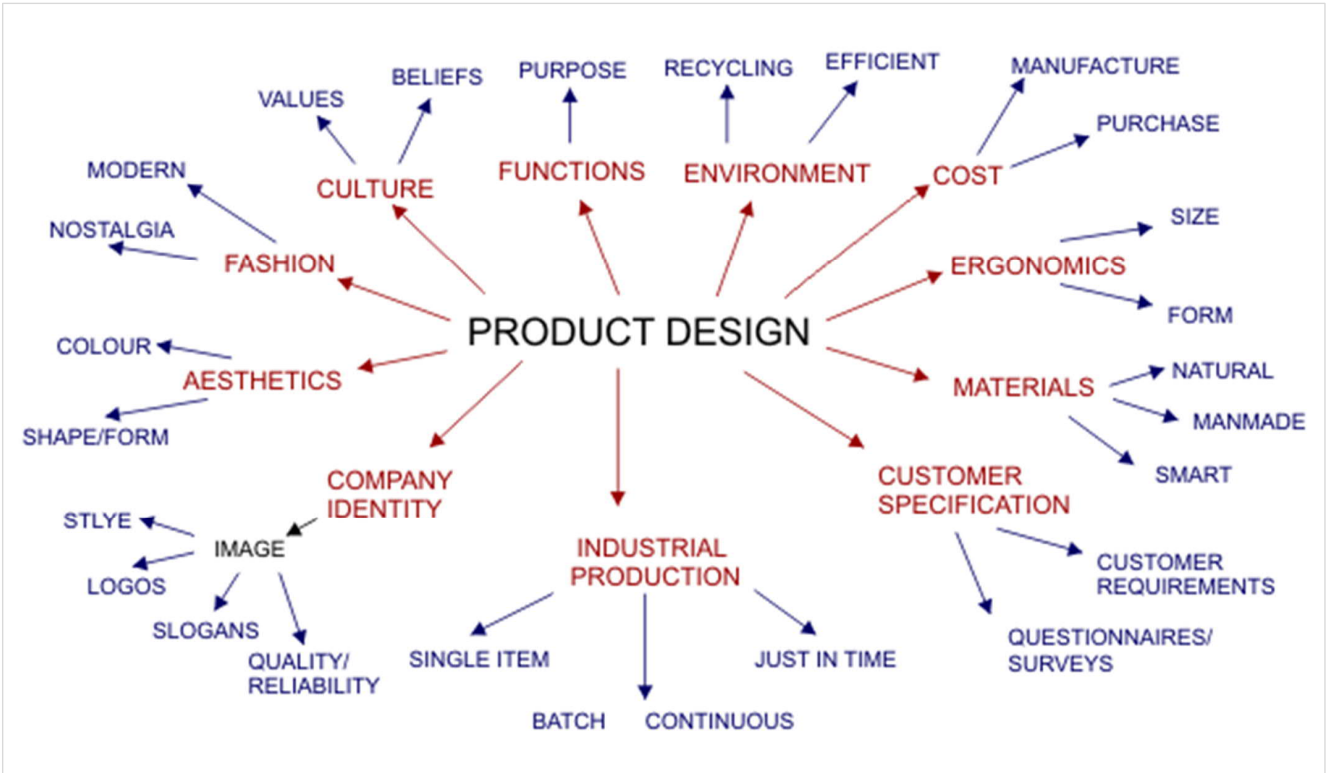
Tampereen yliopisto
Tampere University

Current at course (w 38)

- WE3 this week was: feasibility study and stakeholders
- WE4 next week: requirements

- not all Trello boards are ready yet... invite assistant
- smart groups had group number at Trello board name

- EXAM 1/3 at weeks 40-41; general sw eng, requirements.



[http://dnt-thoughts.blogspot.com/2010/06/development-update.html]

Feasibility study / preliminary analysis

(Feasibility Study was at L2, because of WE3)

Feasibility Study, Preliminary Analysis

- usually made by the vendor/developer (sw company)
- is made at the very beginning, after the idea of a new system
- a few hours ... a few working days is spent
- is made to find out whether a project will be started, or not
- do we have enough time to do such a project, skilled workers, tools, money, possible other/further customers in sight,...
- in many cases the result is that such a project is not started (perhaps later e.g. when technology will be more mature)
- if project is started, matters of Feasibility Study will be described in more detail in Requirements Specification and Project Plan.
- BTW: the first thing is to check internet if such thing has already been done ! "Do not re-invent the wheel."

Tampereen yliopisto
Tampere University

Start for requirements, one path

- customer has some ideas and thoughts about requirements
- developer company writes first basic requirements (based on customer's data)
- developer makes stakeholder search and analysis (which to include)
- developer (re-)defines basic requirements, GUI sketch
- requirements are discussed between customer and developer
- developer writes more concrete requirements (more details), GUI proto
- end-users are taken into account, GUI proto 2
- requirements and GUI are defined in detail
- requirements are discussed in detail between customer and developer
- requirements specification document (includes GUI)

...requirements alternate / interchange / vary at least slightly during the project...

Tampereen yliopisto
Tampere University

Requirements elicitation

Tampereen yliopisto
Tampere University

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3439 requirements elicitation

1. process through which the acquirer and the suppliers of a system discover, review, articulate, understand, and document the requirements on the system and the life cycle processes [ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering, 4.1.18]

2. use of systematic techniques, such as prototypes and structured surveys, to proactively identify and document customer and end-user needs.

3.3447 requirements specification

1. document that specifies the requirements for a system or component cf. design description, functional specification, performance specification

Note 1 to entry: Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards.

Tampereen yliopisto
Tampere University

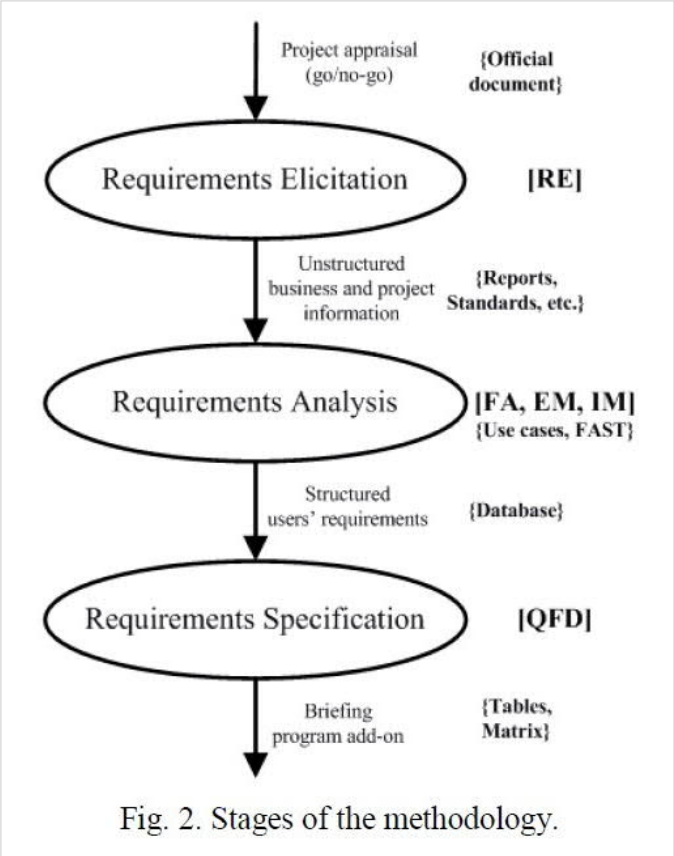


Fig. 2. Stages of the methodology.

[Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010]

RE = Requirements Engineering

FA = Functional Analysis

IM = Information Modelling

FAST = Functional Analysis System Techniques

QFD = Quality Function Deployment

Tampereen yliopisto
Tampere University

How to find ("weed out") requirements

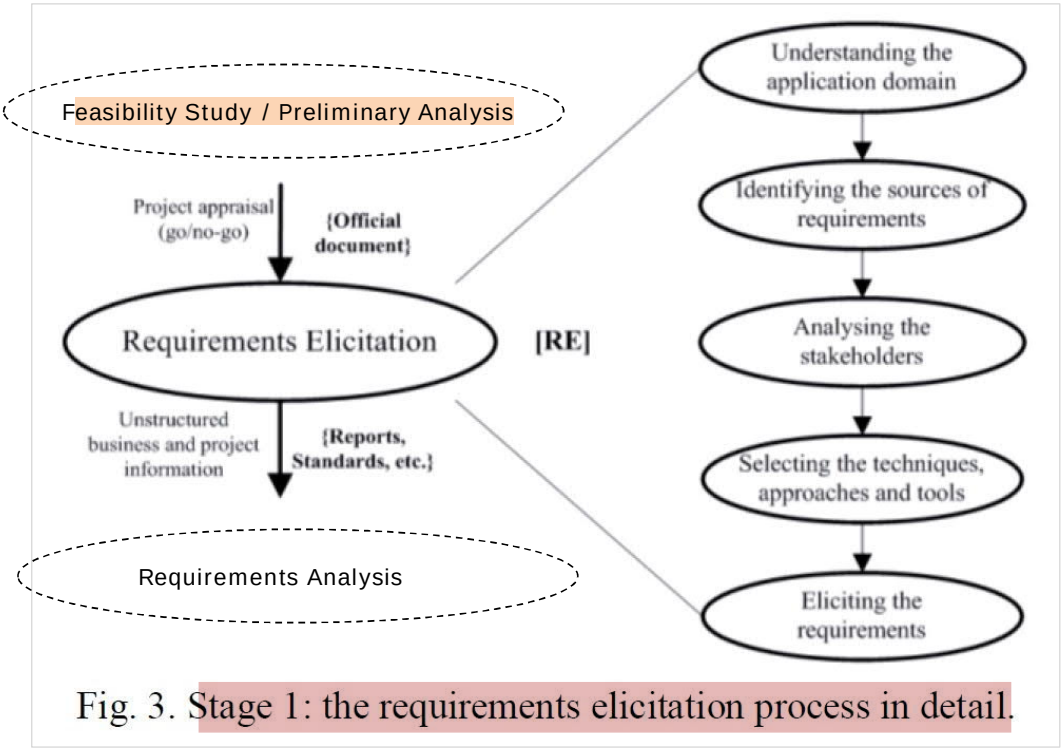


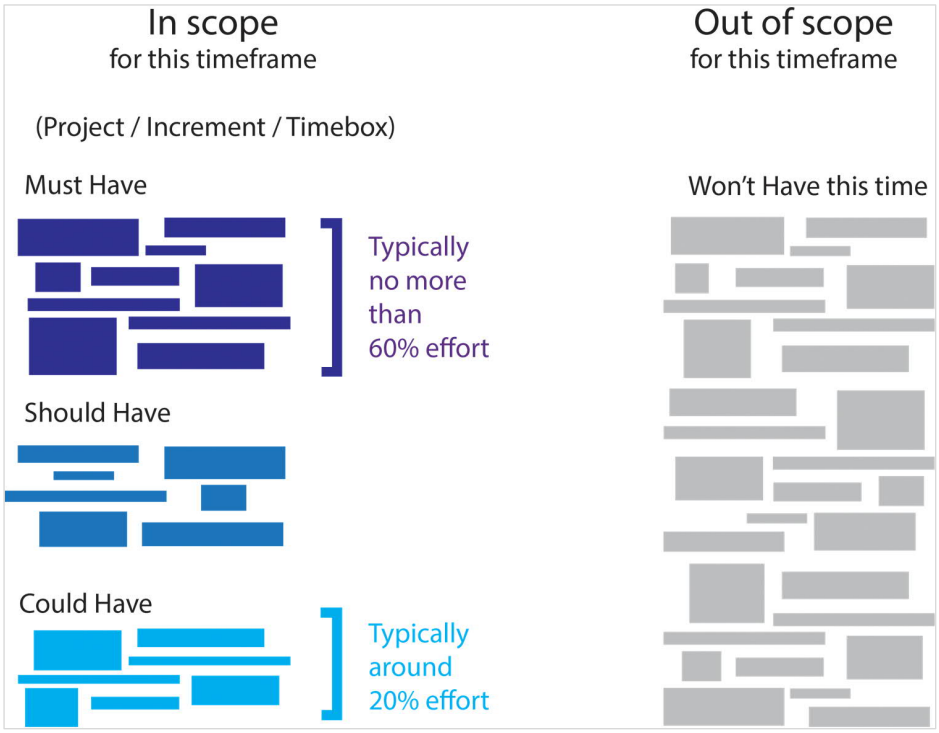
Fig. 3. Stage 1: the requirements elicitation process in detail.

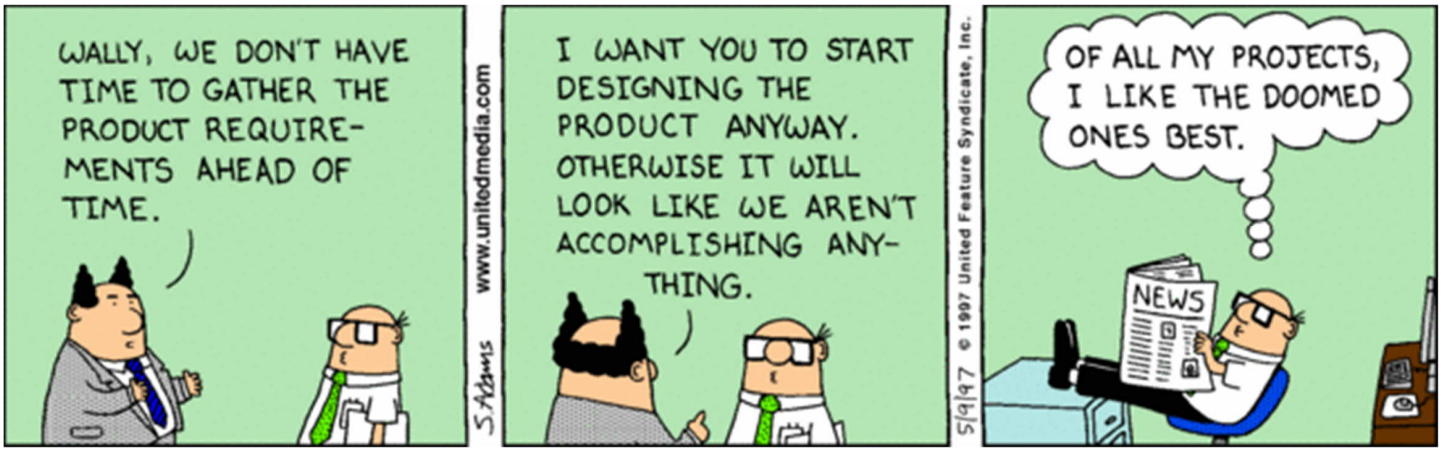
[Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010]

https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation

Sw project features or functionalities; MoSCoW

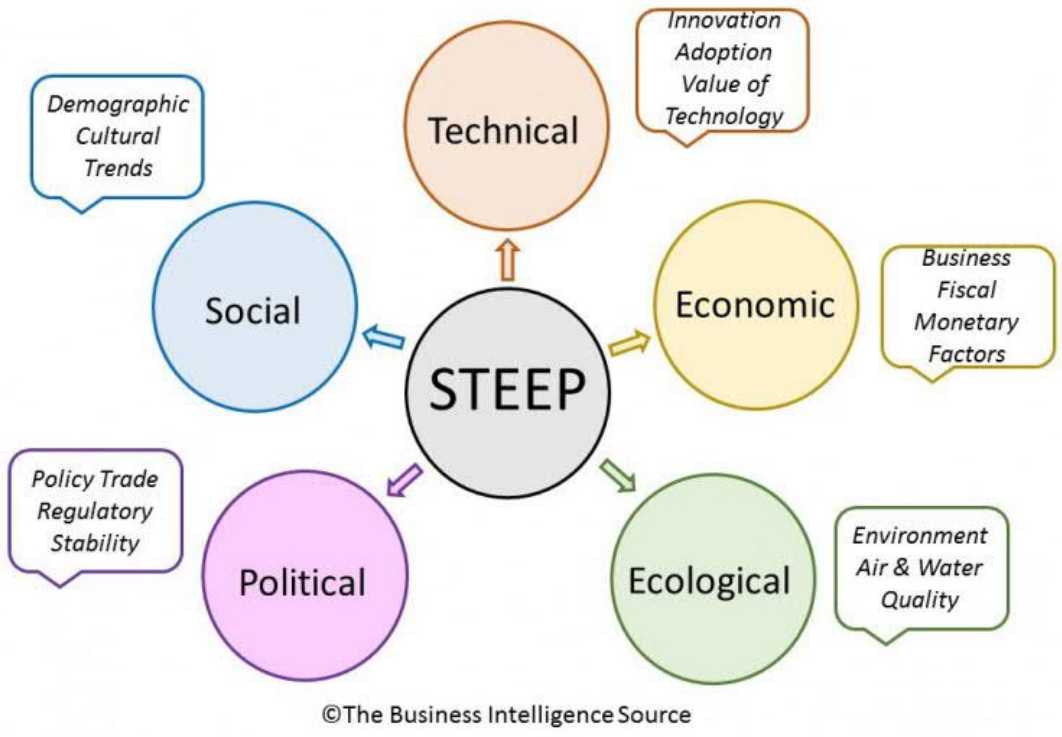
- **M**ust Have
- **S**hould Have
- **C**ould Have
- **W**on't Have this time.





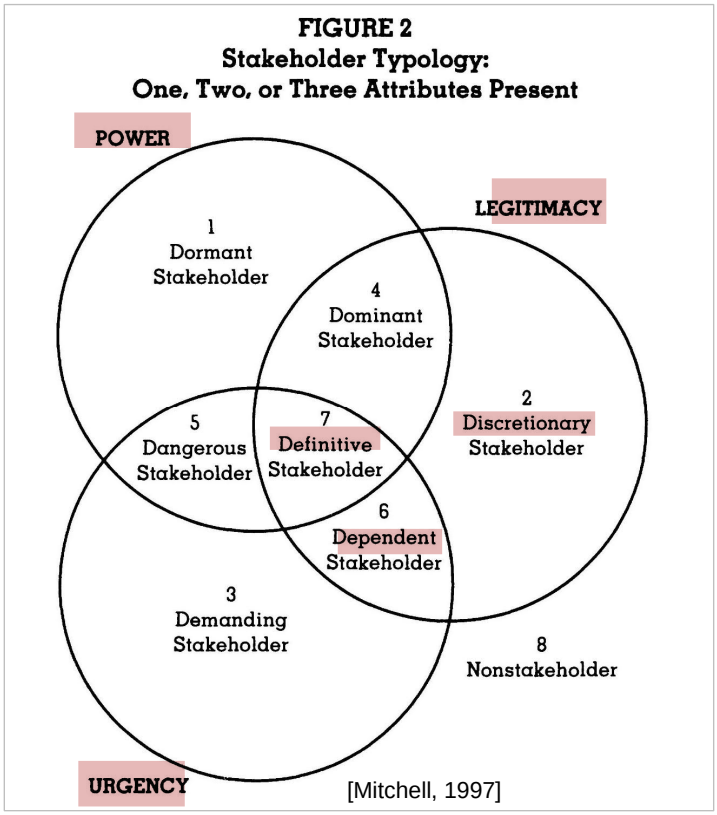
In every sw project, there should be "clear and present" business need.
Sw projects should not be started just because competiting company has such, or just because "we can".
ICT should help companies and people, and to make life better.

Stakeholders



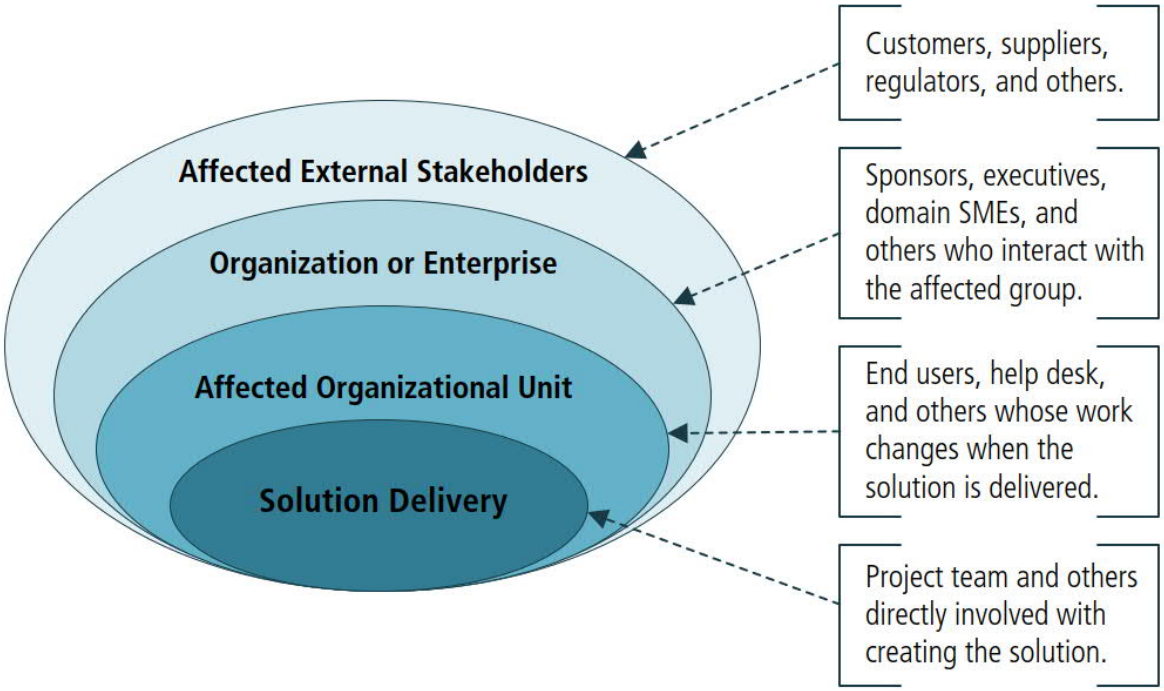
Stakeholders

”Important stakeholders are all who have possibilities and will to affect the project.”



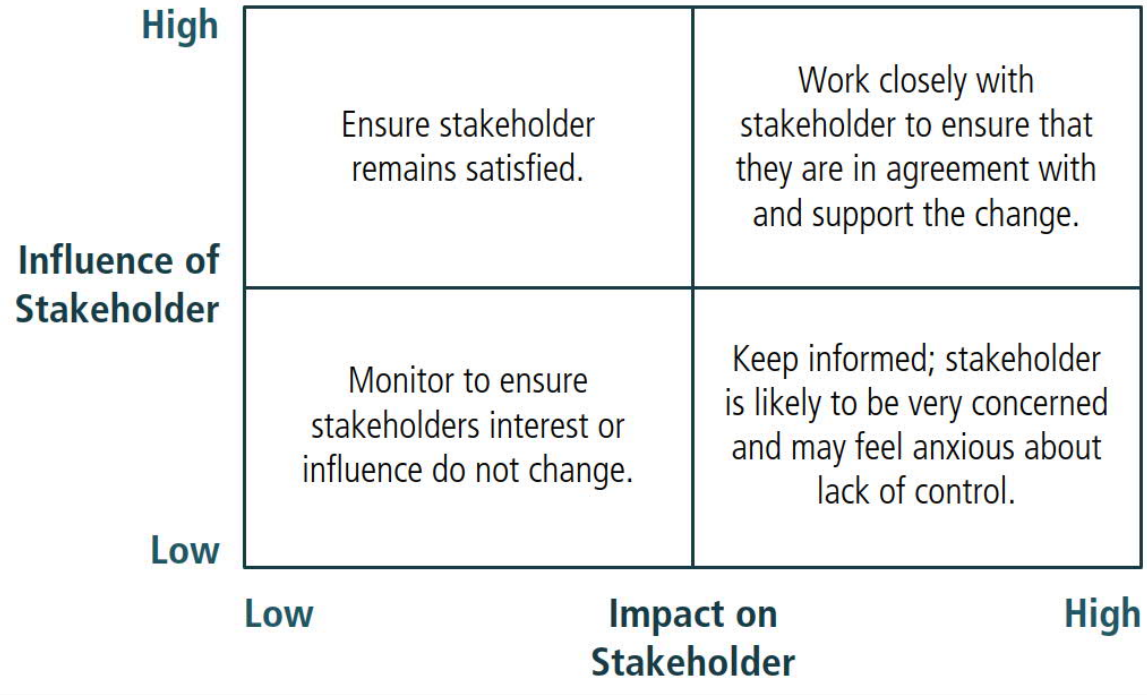
[Mitchell, 1997]

Figure 10.43.2: Stakeholder Onion Diagram



[BABOK v3, 2015]

Figure 10.43.1: Stakeholder Matrix

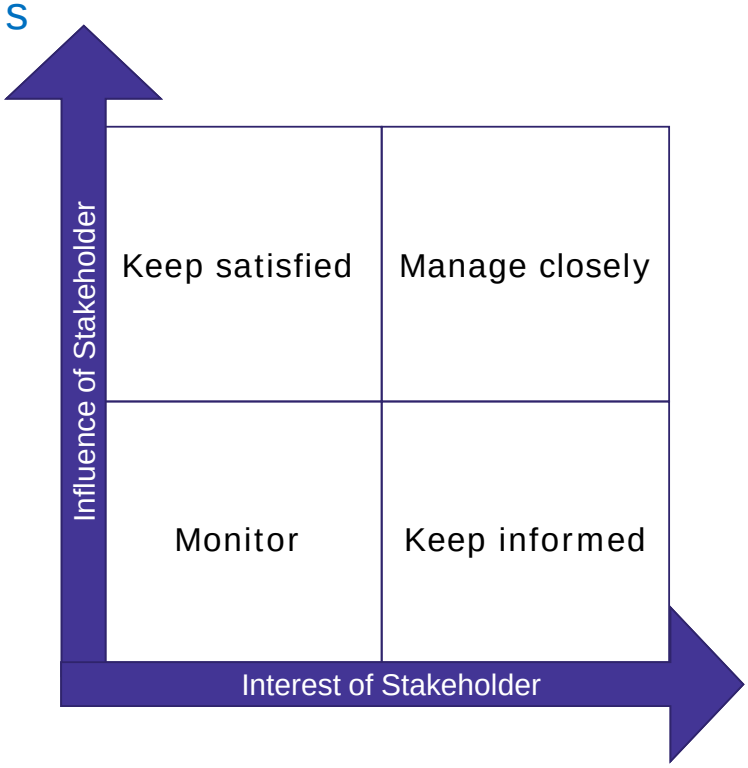


[BABOK v3, 2015]

Stakeholders

At this matrix top right corner surely contains the most important stakeholders.

But depending on the project, also other groups may and should be taken into account, when finding out requirements.



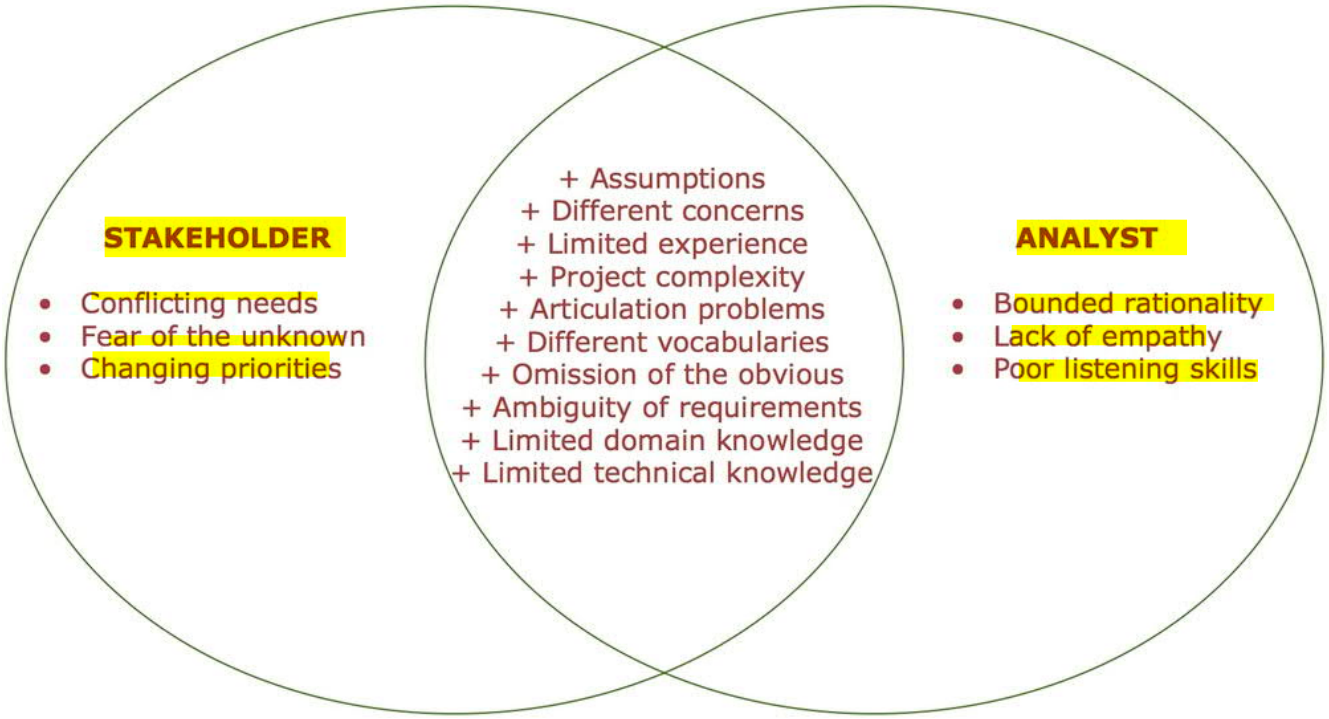
PESTLE… STEEPLE… stakeholders

The following illustrates the PESTLE analysis of Microsoft dated back in 2011:

Table 1-2: PESTLE analysis of Microsoft in 2011

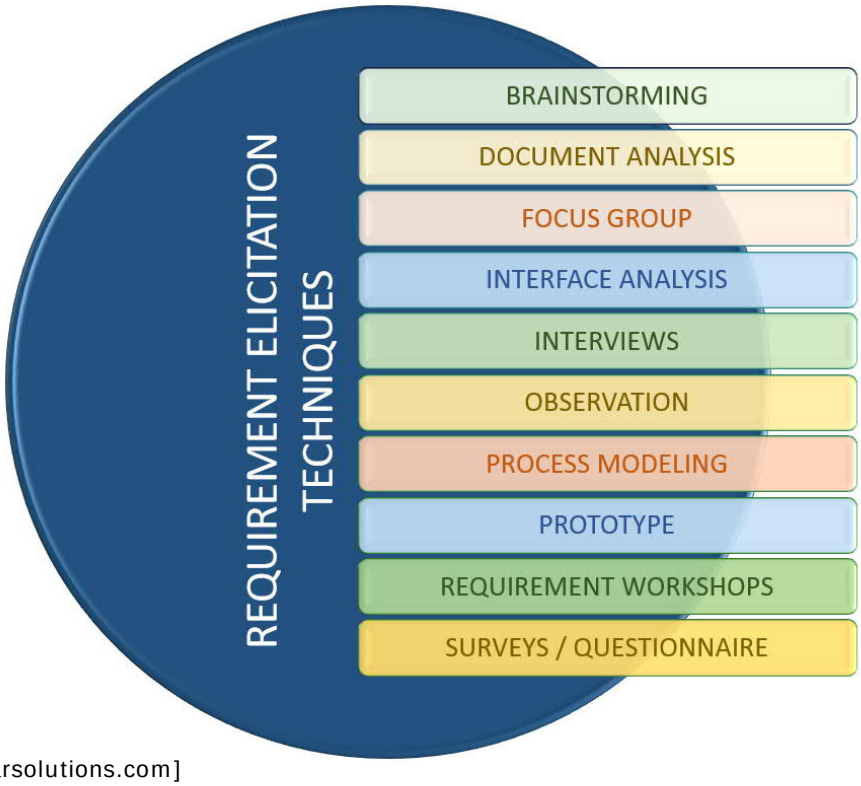
Political • Currency and taxation policies in different countries could significantly increase cost of business operations. • Weak intellectual property protection in emerging markets impairs revenue from such markets.	**Economic** • After-effects of the 2008 economic crisis affect customer demand. • Exchange rate fluctuation makes global pricing strategies more difficult.	**Social** • Increasing awareness of open technologies and business ethics impair the reputation of the Microsoft brand. • Lack of mobile presence causes people to relate Microsoft as an outdated and less innovative brand.
Technological • Mobile, cloud-based and service-based services are gaining more popularity but Microsoft's offerings are not good enough. • Time-to-market of competitors has been shortened significantly. • Competitors' solutions do better on end-user usability and "coolness" (e.g. Apple iPhone)	**Legislative** • Legal disputes on monopoly and abusive practices. • Legal disputes on patent infringements. • Privacy concerns on cloud-based products.	**Environmental** • Power consumption of data centres • Eco-concerns on product packaging materials.

A Snapshot Of The Difficulties Inherent In Requirements Elicitation



[A beginner's guide to requirements elicitation, 2015]

Some requirement elicitation techniques



[www.anarsolutions.com]

2.2 Software Architecture Description

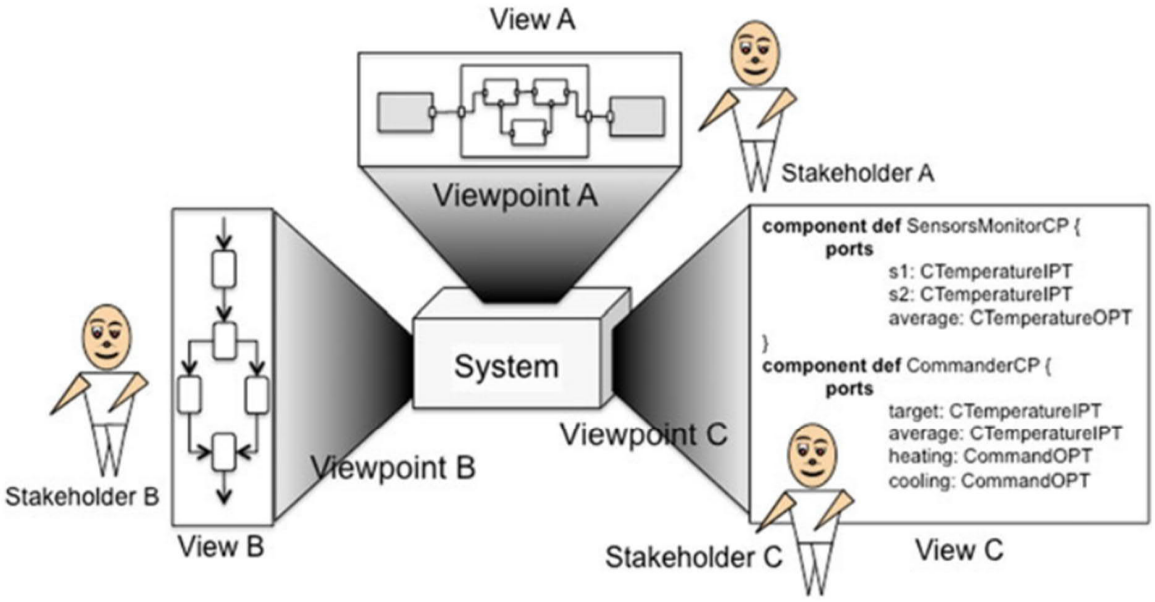


Fig. 2.5 Viewpoints and views

[Software Architecture in Action, 2016]

Tampereen yliopisto
Tampere University

One useful table in SRS may be "permissions matrix"

[BABOK v3, 2015]

Figure 10.39.1: Roles and Permissions Matrix

Roles and Permissions Matrix	Role Group 1			Role Group 2		
Activity		Administrator	Manager		Sales	Customer
Create new account		X	X			X
Modify account		X	X			X
Create order		X	X		X	X
View reports		X	X		X	
Create reports		X	X		X	

Requirements

requirements, three main cases

Functional requirements , e.g.

- user's functionalities
- inputs and outputs
- browser Front and Back buttons are disabled

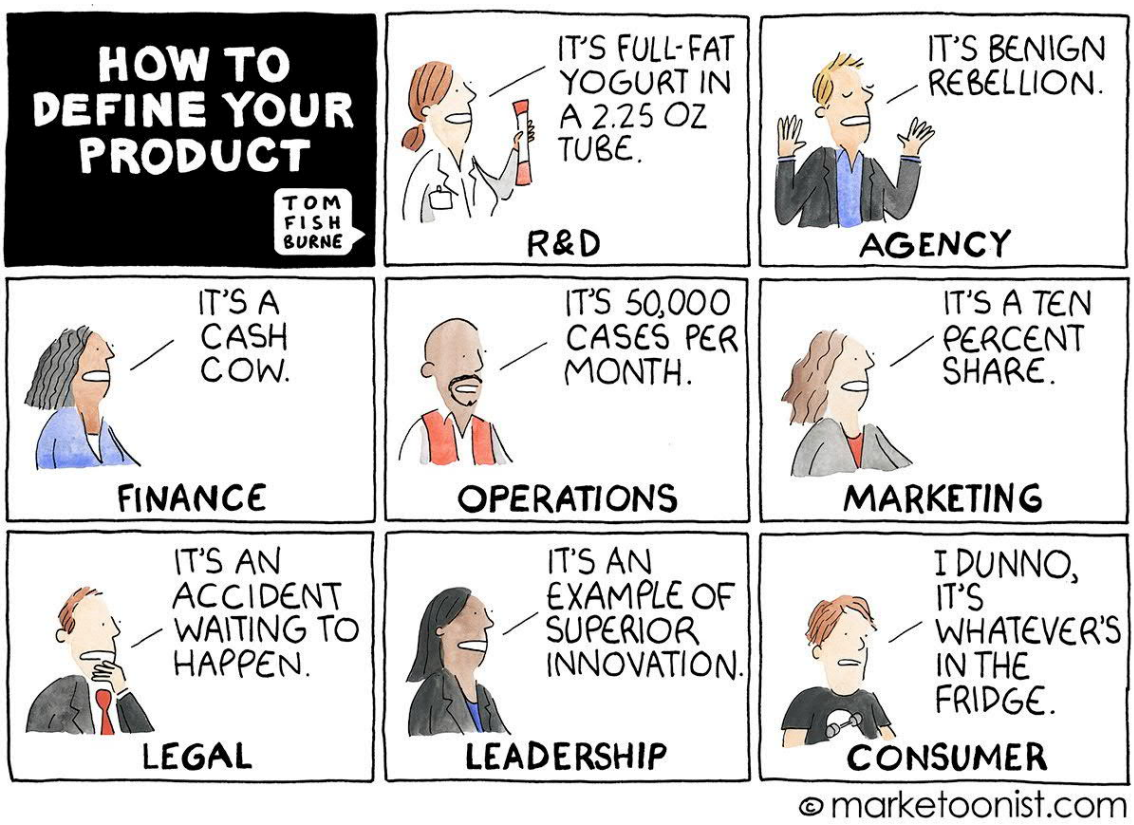
Non-functional requirements , e.g.

- GUI is as customer's guidebook XYZ-2019-A, coding conventions
- localisation, internationalisation = I(18)N
- safety and security

Restrictions and limitations , e.g.

- compatible with customer's old system (DB)
- supported operating systems and browsers
- project schedule and budget.

In some cases a requirement may be considered as non-functional or restriction.



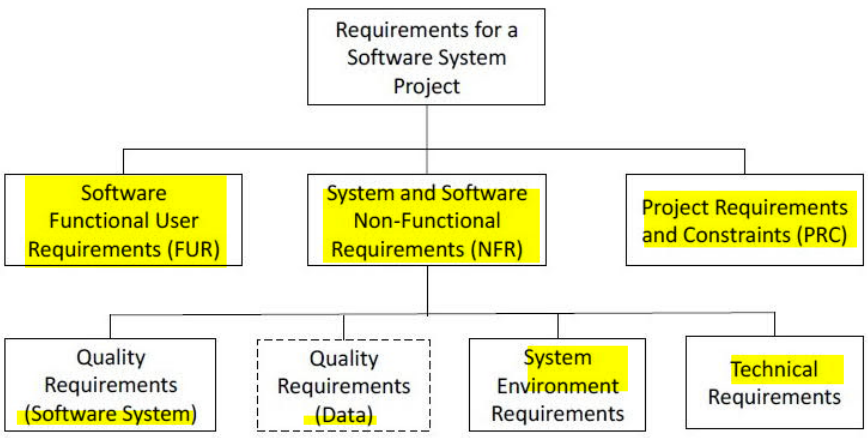
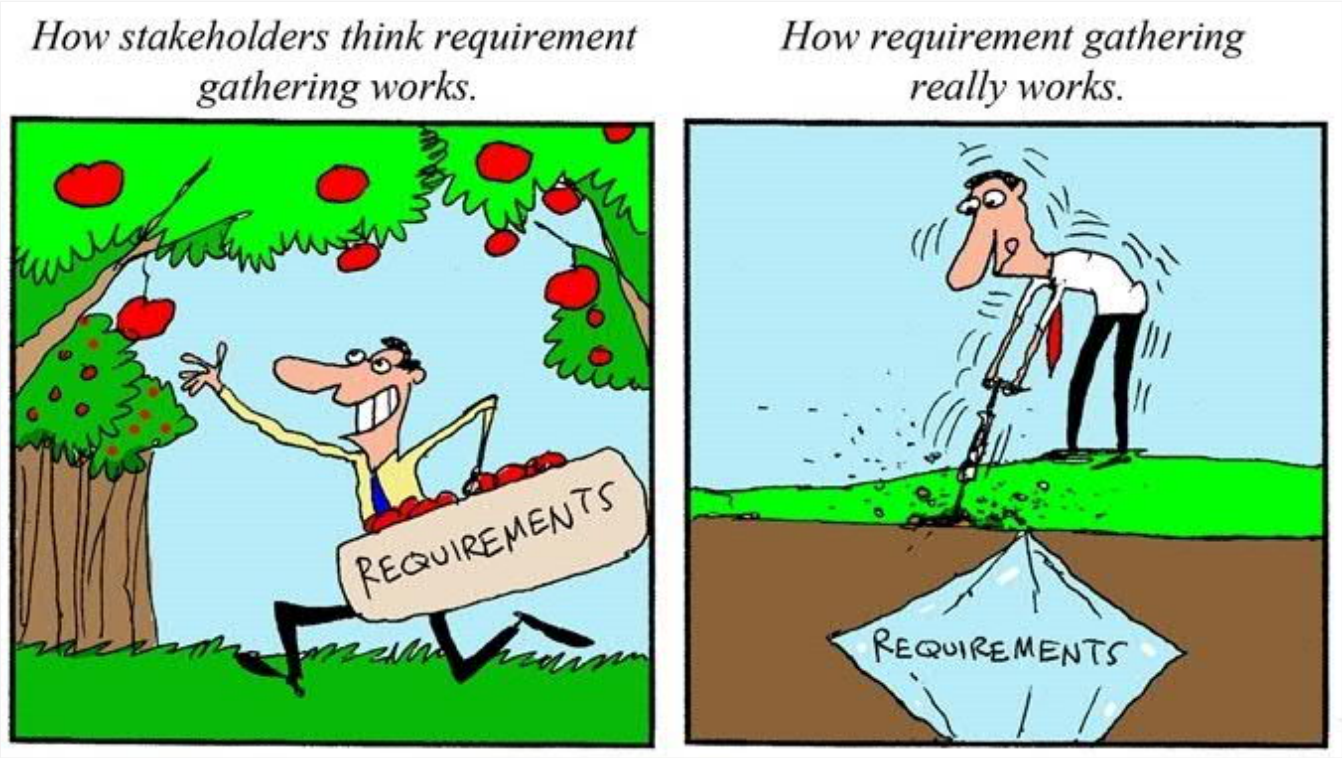


Figure 2.1 - Summary model of requirements for a software systems project



ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3431 requirement

1. statement that translates or expresses a need and its associated constraints and conditions [ISO/IEC TS 24748-1:2016 Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management, 2.41; ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering, 4.1.19]

2. condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents [IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2]

3. provision that contains criteria to be fulfilled [ISO/IEC 14143-2:2011 Information technology — Software measurement — Functional size measurement — Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1, 3.10]

4. a condition or capability that must be present in a product, service, or result to satisfy a contract or other formally imposed specification [A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition] cf. design requirement, functional requirement, implementation requirement, interface requirement, performance requirement, physical requirement.

EXAMPLE: software component requirement

Tampereen yliopisto
Tampere University

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3434

requirements analysis

1. process of studying user needs to arrive at a definition of system, hardware, or software requirements

2. process of studying and refining system, hardware, or software requirements

3. systematic investigation of user requirements to arrive at a definition of a system [ISO/IEC 2382:2015, Information technology — Vocabulary]

4. determination of product- or service-specific performance and functional characteristics based on analyses of customer needs, expectations, and constraints; operational concept; projected utilization environments for people, products, services, and processes; and measures of effectiveness.

Tampereen yliopisto
Tampere University

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3895 , specification

1. detailed formulation, in document form, which provides a definitive description of a system for the purpose of developing or validating the system *[ISO/IEC 2382:2015, Information technology — Vocabulary]*

2. Information item that identifies, in a complete, precise, and verifiable manner, the requirements, design, behavior, or other expected characteristics of a system, service, or process *[ISO/IEC/IEEE 15289:2015 Systems and software engineering — Content of life-cycle information products (documentation),* 5.24*]*

3. a document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system, component, product, result, or service and, often, the procedures for determining whether these provisions have been satisfied. Examples are: requirement specification, design specification, product specification, and test specification *[A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition]*

4. Concrete representation of a model in some notation *[ISO/IEC 10746-2:2009 Information technology — Open Distributed Processing — Reference Model: Foundations,* 7.4*].*

Tampereen yliopisto
Tampere University

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.4089

SyRS

1. System Requirement Specification [ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering, 4.2]

cf. SRS.

Requirements traceability is needed only in ”mission-critical” systems, e.g. satellite systems and healthcare machines. There are special software for requirements management.

Tampereen yliopisto
Tampere University

Requirements... how to see those

In such cases negotiations are needed...

ISO 24765:2017 Systems and software engineering — Vocabulary

3.1582 feature (FI: ominaisuus)

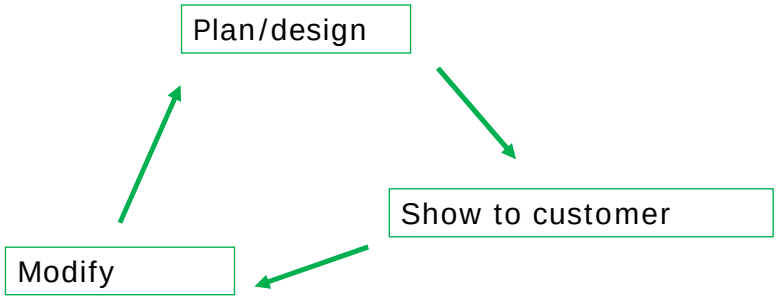
- 1. distinguishing characteristic of a system item
- 2. functional or non-functional distinguishing characteristic of a system, often an enhancement to an existing system
- 3. abstract functional characteristic of a system of interest that end-users and other stakeholders can understand

3.1716 functionality (FI: toiminnallisuus)

- 1. capabilities of the various computational, user interface, input, output, data management, and other features provided by a product
- Note 1 to entry: This characteristic is concerned with what the software does to fulfill needs. The software quality characteristic functionality can be used to specify or evaluate the suitability, accuracy, interoperability, security, and compliance of a function.

start quickly, small steps, iterate

- When dealing with customer who doesn't actually know exactly what (s)he wants, make demos/prototypes quickly and often.
- When showing first hand-written sketches of GUI to customer after two weeks, you always get some comments (yes/no). Then update your demo/requirements and show next version after 1..2 weeks.
- It is better to show your ideas of the software system often to customer, instead of thinking yourself many weeks and fine-tuning a demo you THINK customer wants.
- Work in iterations.



SFS-EN 16603-10-06:2014

6.2 Identification of types of technical requirements

6.2.1 Introduction

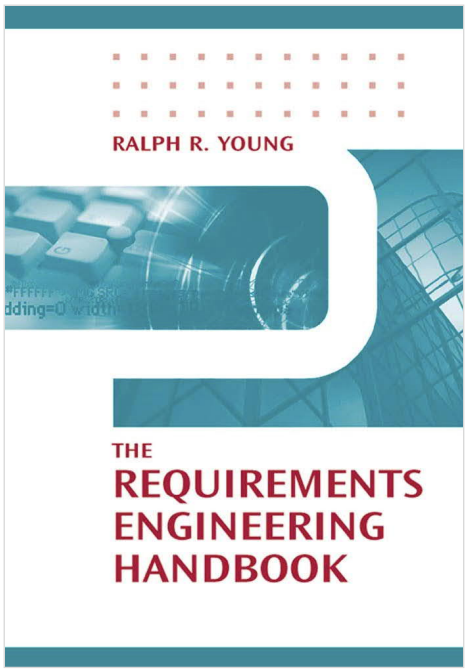
The differing types of technical requirements contained in the TS are as follows

- functional requirements,
- mission requirements,
- interface requirements,
- environmental requirements,
- operational requirements,
- human factor requirements,
- (integrated) logistics support requirements,
- physical requirements,
- product assurance (PA) induced requirements,
- configuration requirements,
- design requirements,
- verification requirements.

NOTE These different technical requirements are called "user related functions" and constraints in EN 1325-1.

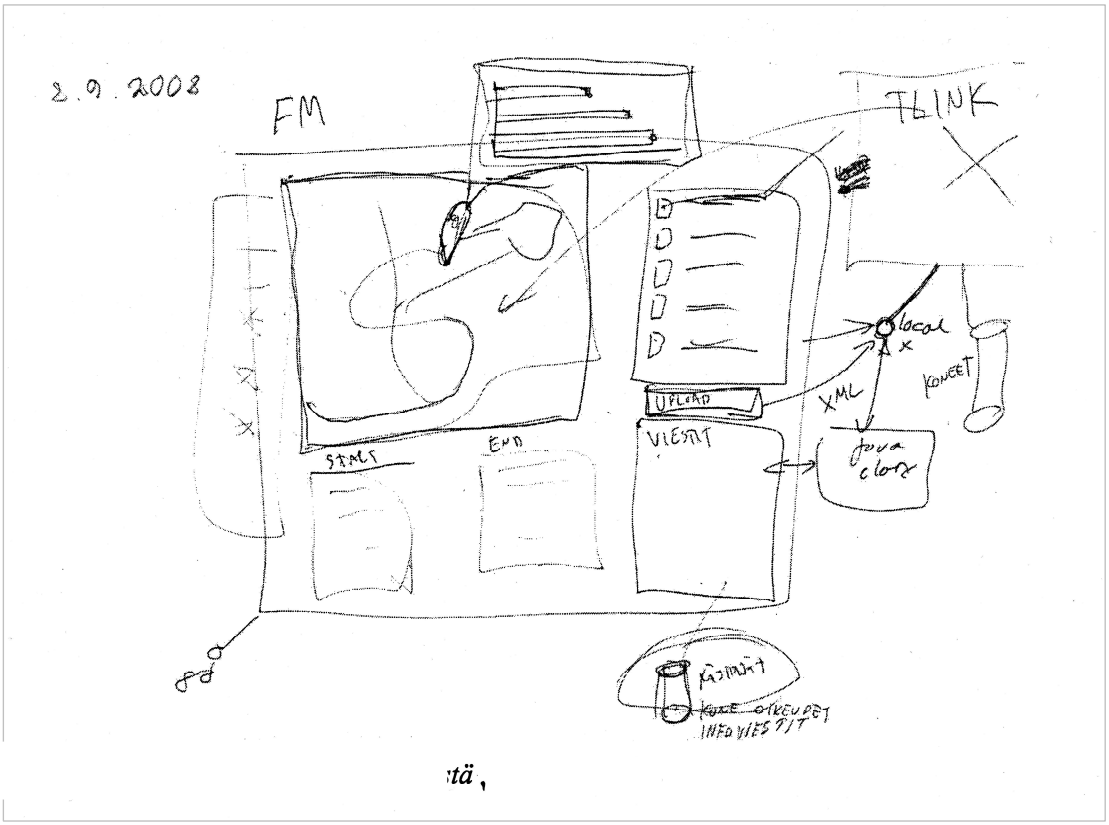
Definitions and Descriptions of Requirements Types

- Business Requirements
- Stated Requirements Versus Real Requirements
- User Requirements
- High-Level or System-Level Requirements
- Business Rules
- Functional Requirements
- Nonfunctional Requirements
- Derived Requirements
- Design Requirements and Design Constraints
- Performance Requirements
- Interface Requirements
- Verified Requirements
- Validated Requirements
- Qualification Requirements
- The “Ilities” and Specialty Engineering Requirements
- Unknowable Requirements
- Product Requirements
- Process Requirements
- Logistics Support Requirements
- Environmental Requirements
- System, Subsystem, and Component Requirements.



Typical first meeting memo from developer-customer meeting; first view of the planned system.

But no pictures nor text alone are enough, both are needed for complete requirements.



Which car radio would user prefer... ?
Which car radio would developer prefer... ?

10 functions, 100 €

30 functions, 300 €

KISS = Keep It Simple Stupid

Variants of KISS, the KISS principle is also offered in two other forms:
Keep it short and simple and **Keep it simple and straightforward**.

KISS is related to a fair number of other famous quotes, phrases and principles. Among them:

Occam's Razor - ***"Among competing hypotheses, the one with the fewest assumptions should be selected"*** (but often restated as "The simplest solution is most likely the correct solution" which is not quite the same thing).

Albert Einstein's ***– "Make everything as simple as possible but not simpler"***.

Leonardo Da Vinci's – ***"Simplicity is the ultimate sophistication"***.

Ludwig Mies Van Der Rohe's – ***"Less is more"***

Bjarne Stroustrup's ***"Make Simple Tasks Simple!"*** (Stroustrup is a Danish computer scientist).

Antoin Marie Jean-Baptiste Roger, comte de Saint-Exupery's ***"It seems that perfection is not reached when there is nothing left to add but when there is nothing left to take away."***

[https://www.interaction-design.org/literature/article/kiss-keep-it-simple-stupid-a-design-principle]

Requirements specification

[Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010]

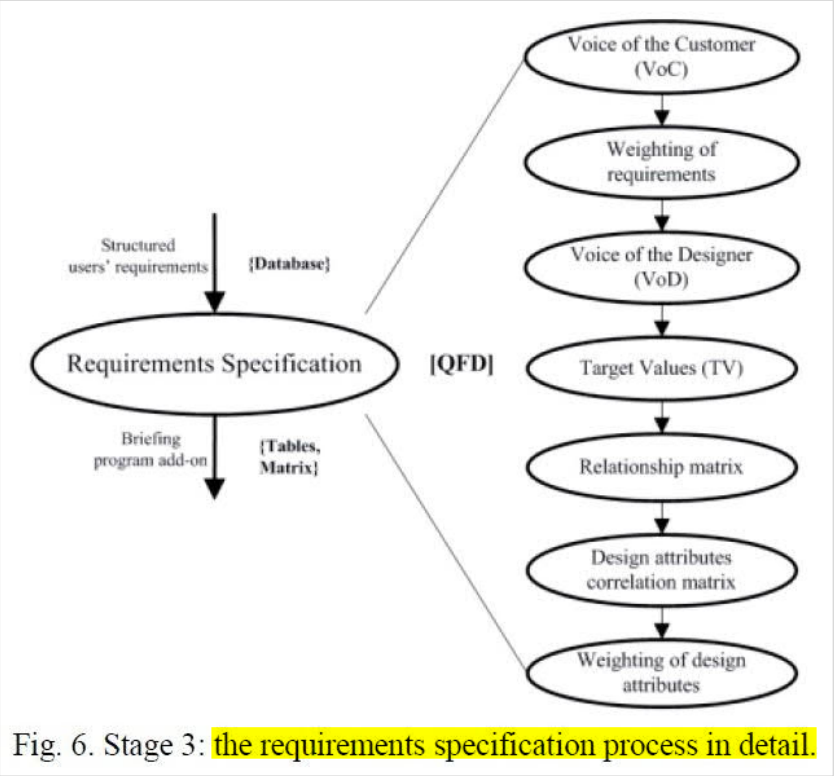
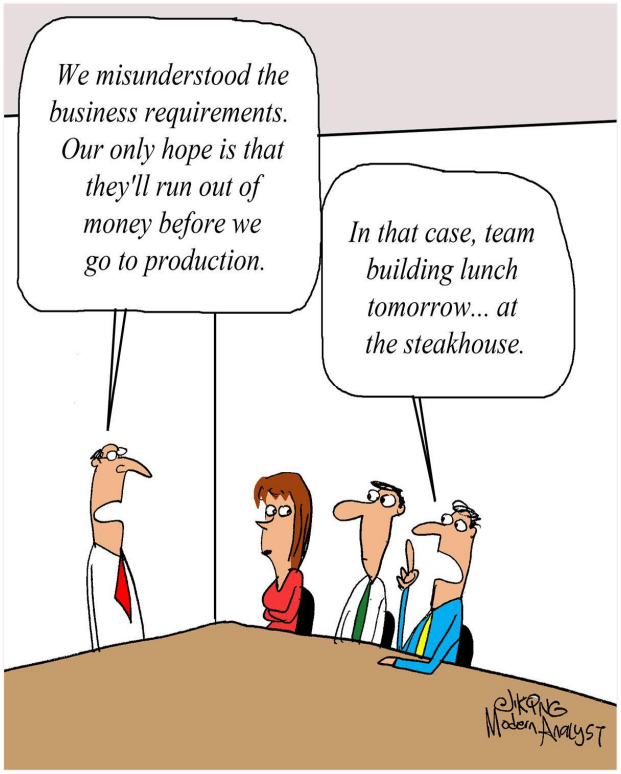
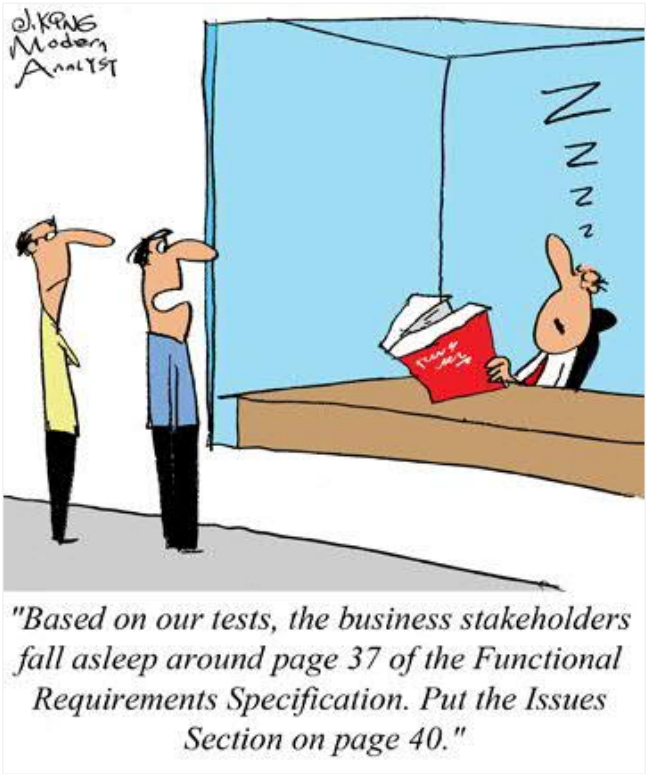


Fig. 6. Stage 3: the requirements specification process in detail.

QFD = Quality Function Deployment

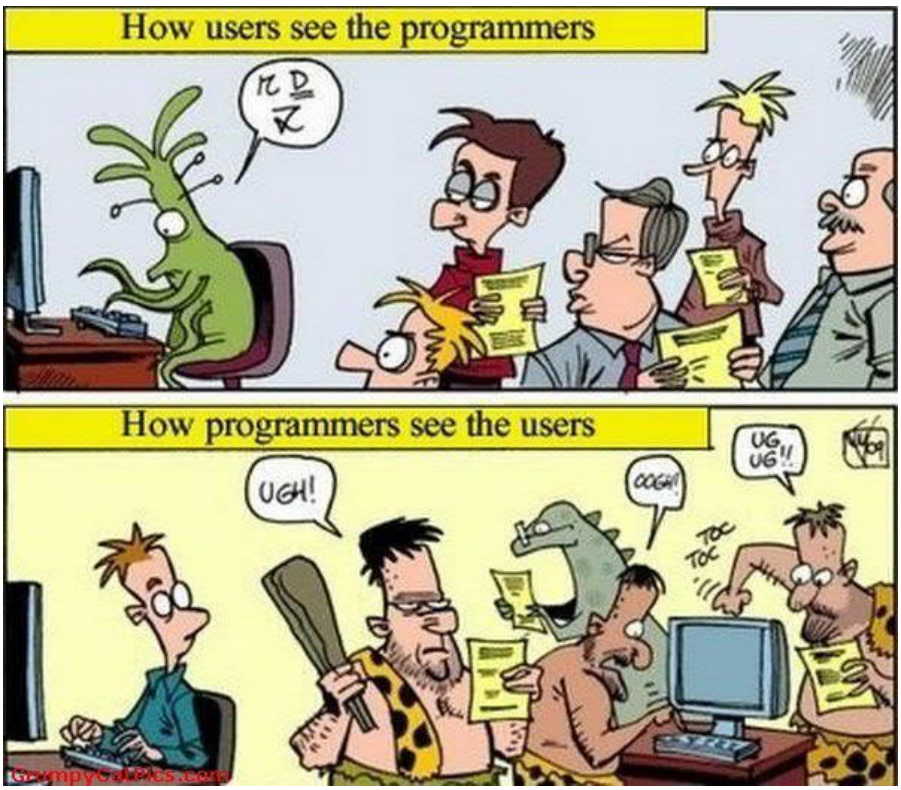




"Based on our tests, the business stakeholders fall asleep around page 37 of the Functional Requirements Specification. Put the Issues Section on page 40."

By the way, remember WHY you make the software, and for WHOM.

”User before technology...?”



Tampereen yliopisto
Tampere University

COMP.SE.100-EN (ItSE)
Introduction to Software Engneering

Lecture 3, 16.09.2020

Tensu: remember to pause Zoom lecture recording

Zoom lecture break, 10 minutes stretching, walking, etc.

Tampereen yliopisto
Tampere University

Contraducting requirements and goals;
- technical goals
- business goals
- user's goals
- etc.

(this is just an example)

The many demands on a product's hardware and software must be traded-off to find a workable solution

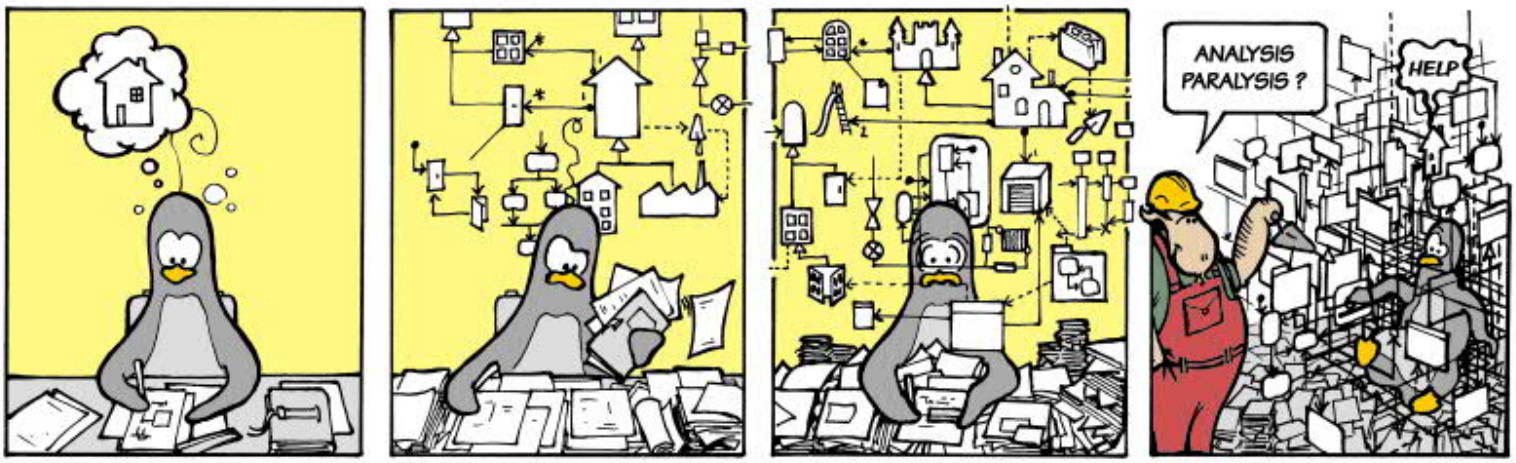
Beware "feature creep".

Agile methods help in this
- start with basic features
- develop and test in iterations
- show demo/proto to customer
- customer gives feedback
- modify demo/proto for next version.

By the way, is it customer's NEEDS or wishes or delusions which are written as requirements ?

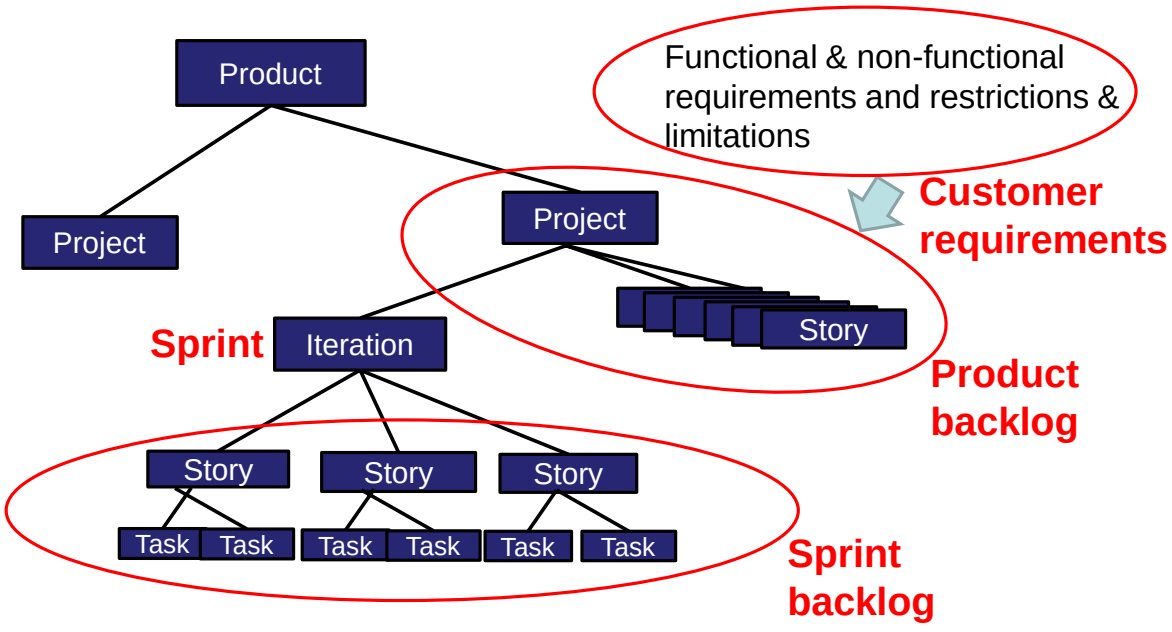
Reality check ?!

Agile methods bring help to ”analysis paralysis”



Feature creep = more and more features are added,
but… project budget and schedule are not updated accordingly.
(FI: paisuu kuin pullataikina)

How do you eat an elephant? In small pieces...



Chris Doig: Five ways inadequate requirements wreak havoc with enterprise software purchases

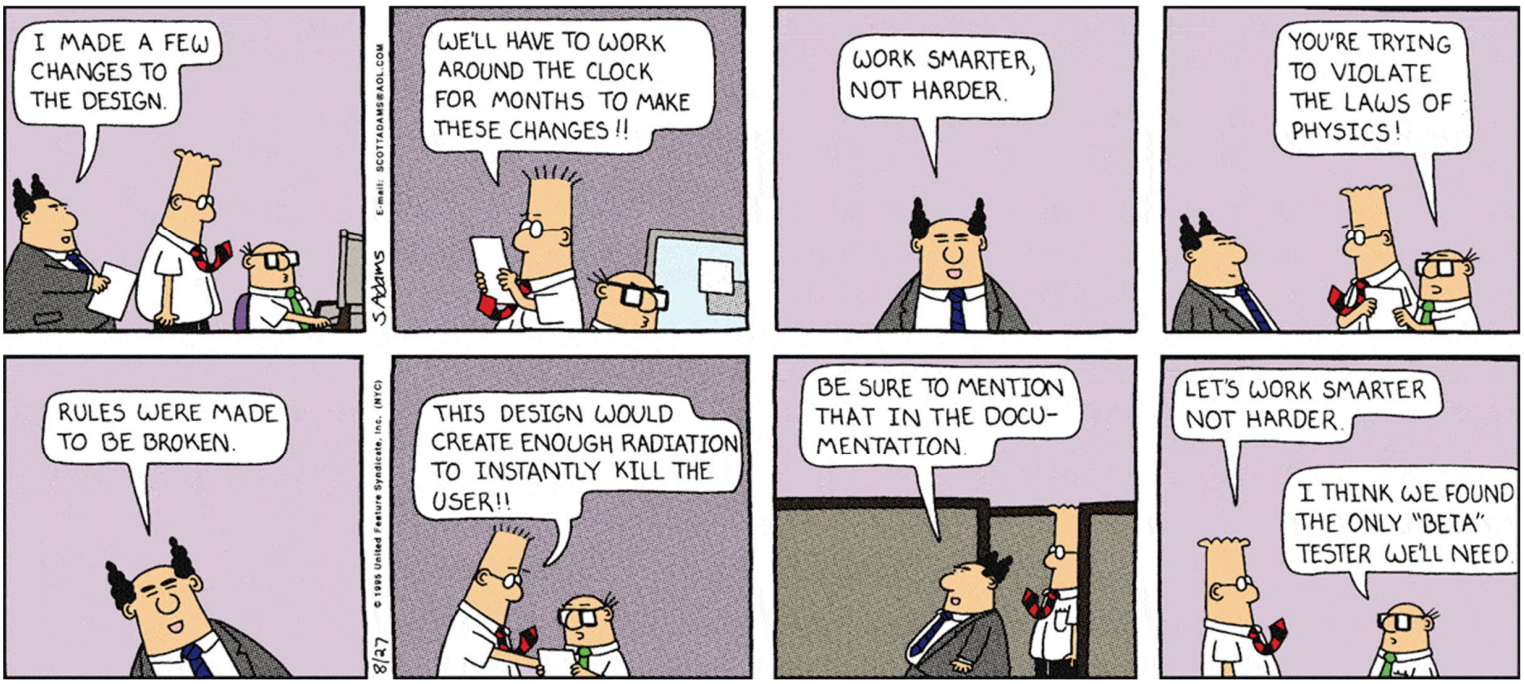
1) Inadequate functionality
2) Discovering “new” requirements
3) Implementing “new” requirements
4) Business disruption
5) Unmet expectations.

Requirements will be discovered during the initial requirements analysis, during the implementation or in early production use. An implementation project that starts with a comprehensive list of ALL significant requirements, who wants them, why they are wanted and how important they are, enables the project manager to create a realistic implementation plan. When no significant new requirements are discovered, the implementation is on schedule and within budget.

[www.cio.com]

Remember to update documentation after changes.

Requirements may / will / surely change...

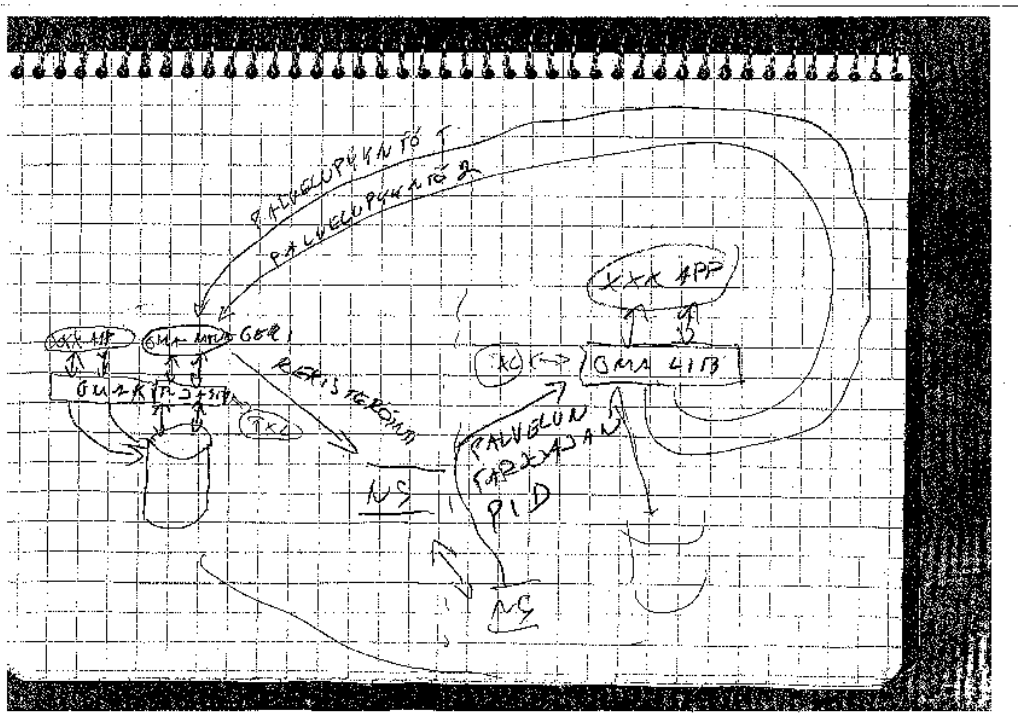


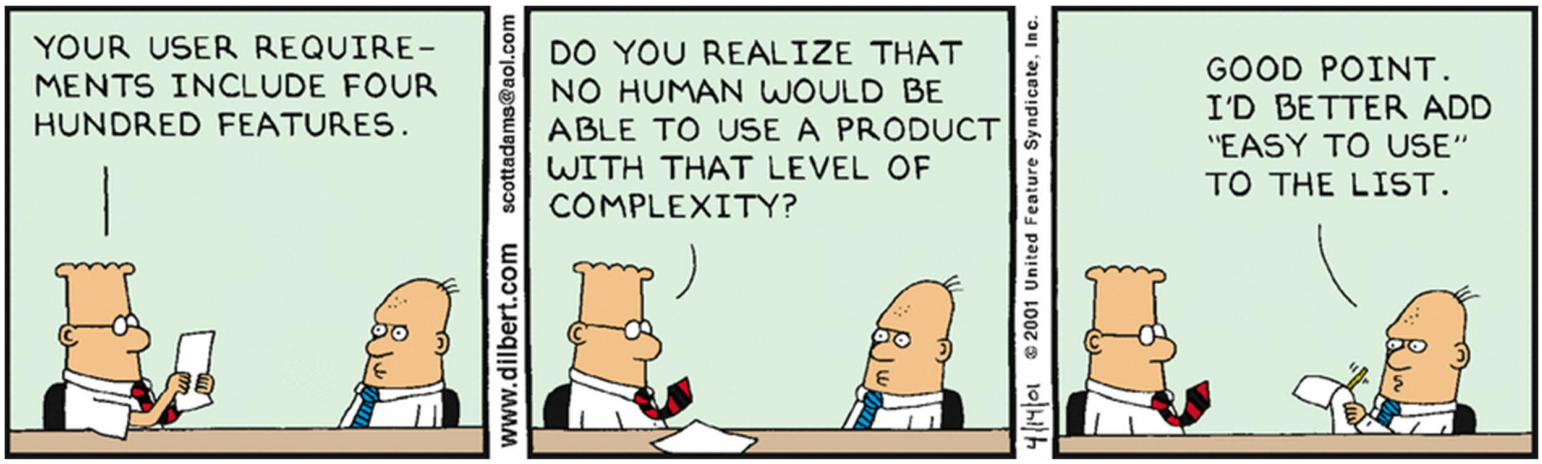
Good specification is

- complete
- accurate, specific
- faultless, soundness
- understandable
- testable (measurable, to fulfill requirement)
- traceable
- no redundancy
- as short as possible.

If there would be mind-reading, requirements documentation would be obsolate.

Memo from a first customer meeting





”Easy to use” = user is cursing software less than once in a minute ?

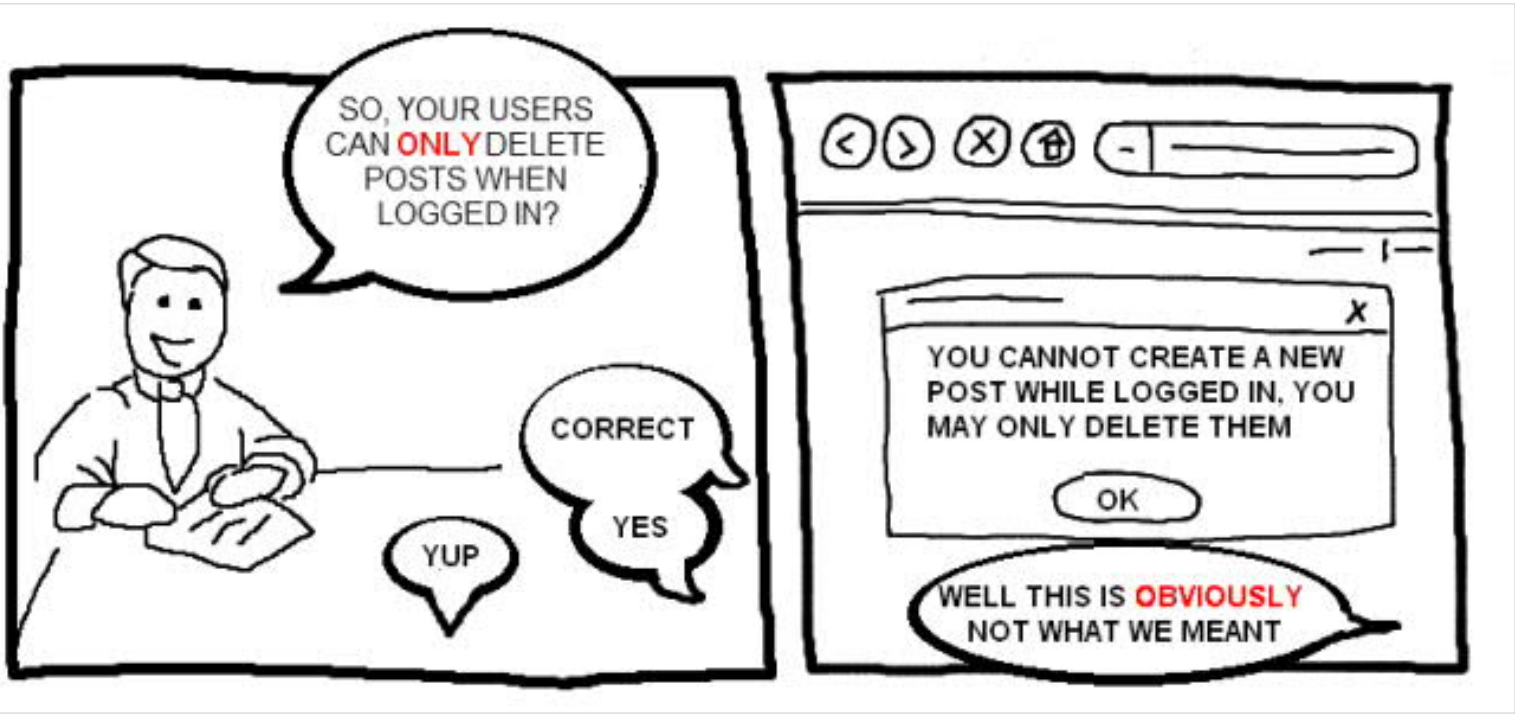
Requirements (TS = technical specification) should be unambiguous/unequivocal

8.3.3 Format restrictions

a. List of terms that shall not be used in a TS requirement (technical requirements specification) “and/or”, “etc.”, “goal”, “shall be included but not limited to”, “relevant”, “necessary”, “appropriate”, “as far as possible”, “optimize”, “minimize”, “maximize”, “typical”, “rapid”, “userfriendly”, “easy”, “sufficient”, “enough”, “suitable”, “satisfactory”, “adequate”, “quick”, “first rate”, “best possible”, “great”, “small”, “large”, and “state of the art”.

[SFS-EN 16603-10-06:2014, SPACE ENGINEERING. PART 10-06: TECHNICAL REQUIREMENTS SPECIFICATION]

Make sure all parties/stakeholders understand requirements correctly



Requirements should be measurable

For example, at preliminary phases customer/user may say that software should be ”user-friendly”. Yes… allright… what does that mean…?

Developer asks what that means, and at next and more defined version of requirements that may be ”easy to use, and user can personalise view”.

At the ”final” version of requirements (at the moment when project starts) it may be written as

- user can do task X with three mouse clicks at most
- user can do task Z in five seconds at most (in average)
- GUI’s response time should be less than 6 seconds at 95 % of cases
- user can change colour theme for GUI (defined in detail later)
- user can hide/reveal additional tools and features panels (defined in detail later)
- user can change locations and sizes of GUI components. (defined in detail later).

Creating User Stories and Tasks

- a (user/customer) requirement may contain one or more User Stories
  - one requirement can affect many user stories
- user story lists:
  - role(s) / actor(s)
  - what is done
  - added value of the story (if not self evident).
- one User Story may be split into one or more tasks
- ideally the size of a task is such, that a developer could finish that during one workday.

Tampereen yliopisto
Tampere University

If requirements change, project plan should cange, too

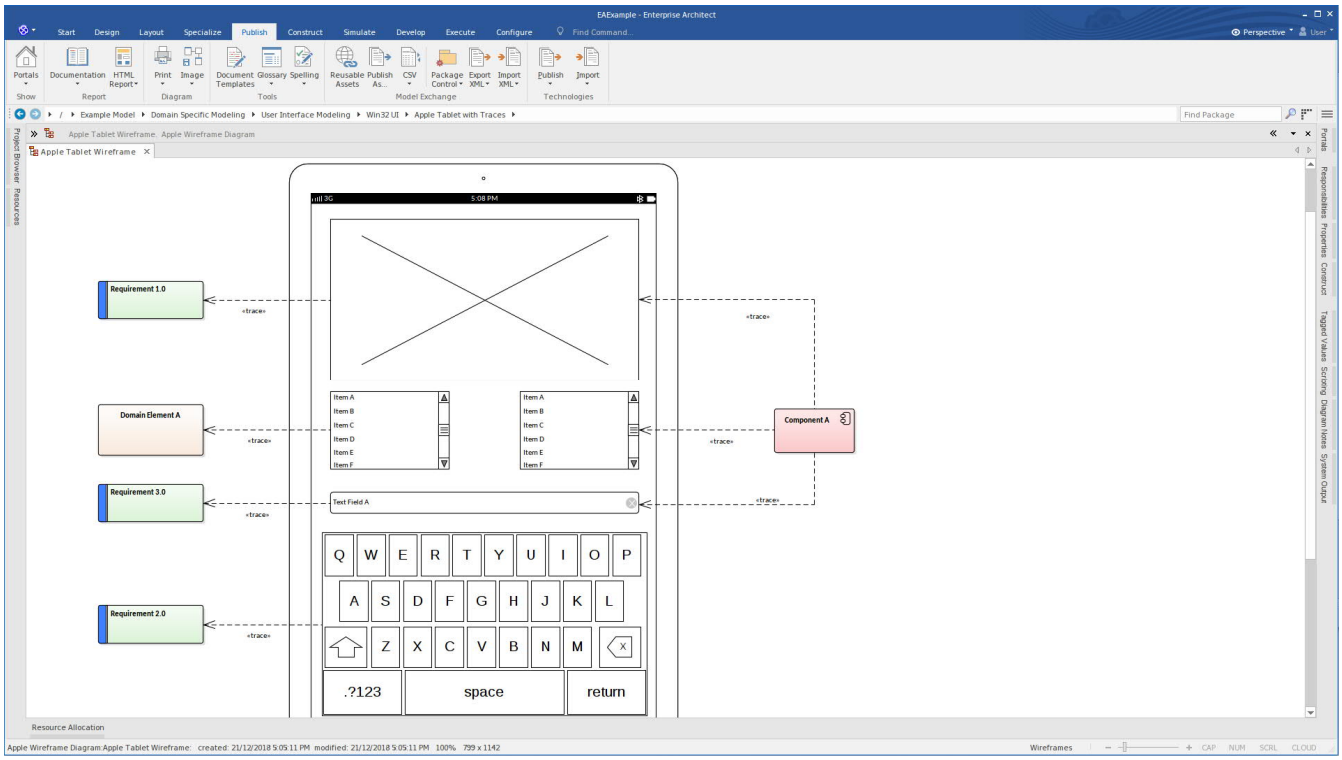
YouTube, by Wiegers, videos 1-35

Tampereen yliopisto
Tampere University

GUI wireframe

Tampereen yliopisto
Tampere University

GUI wireframe

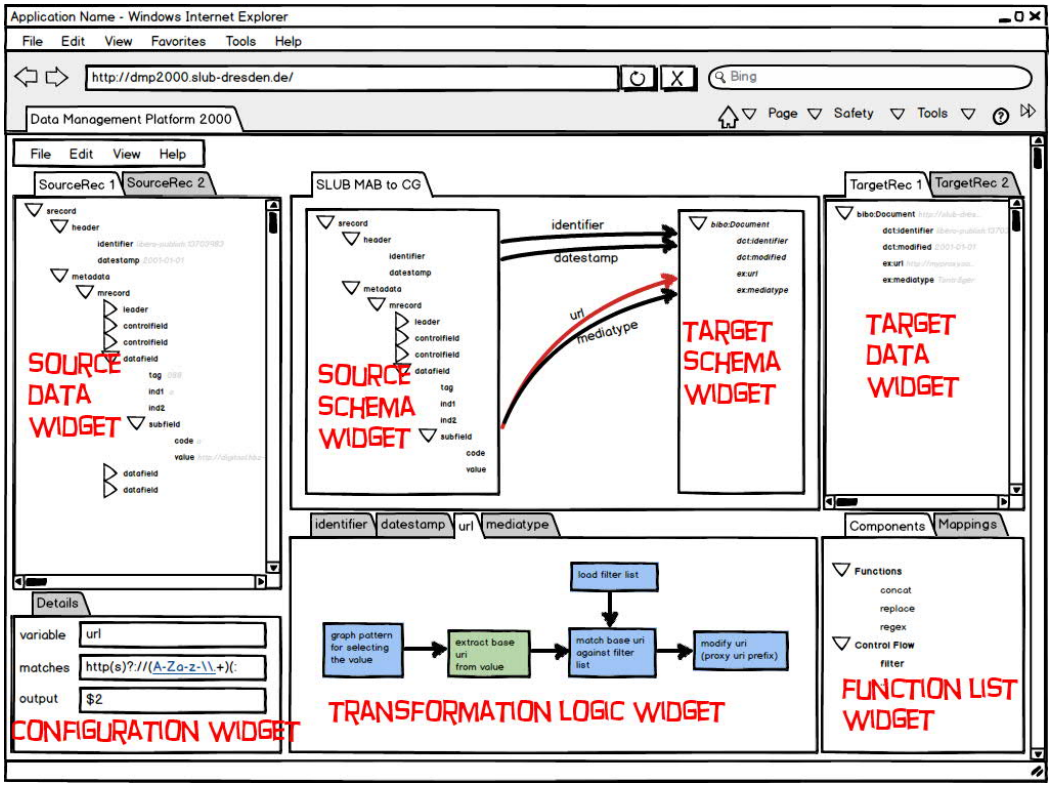


[sparxsystems.com/resources/gallery/diagrams/]

Tampereen yliopisto
Tampere University

[https://webdesignledger.com/wp-content/uploads/2012/10/wireframetools09a.jpg]

GUI wireframe



[https://raw.githubusercontent.com/wiki/dswarm/dswarm-documentation/assets/wireframeGUI.png]

GUI sketching

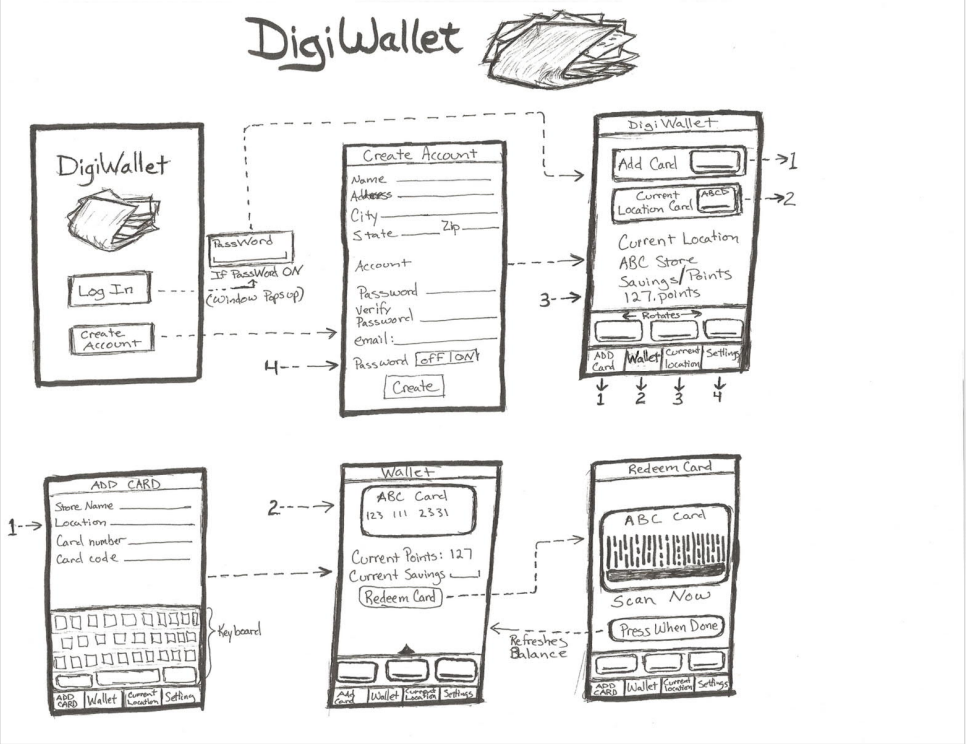
Graphical User Interface (GUI) is often the part where planning and design starts nowadays.

If customer (client) does not know what (s)he wants, draw some GUI sketch (even by free hand) and show that. Surely customer or end-user can comment any kind of drawing.

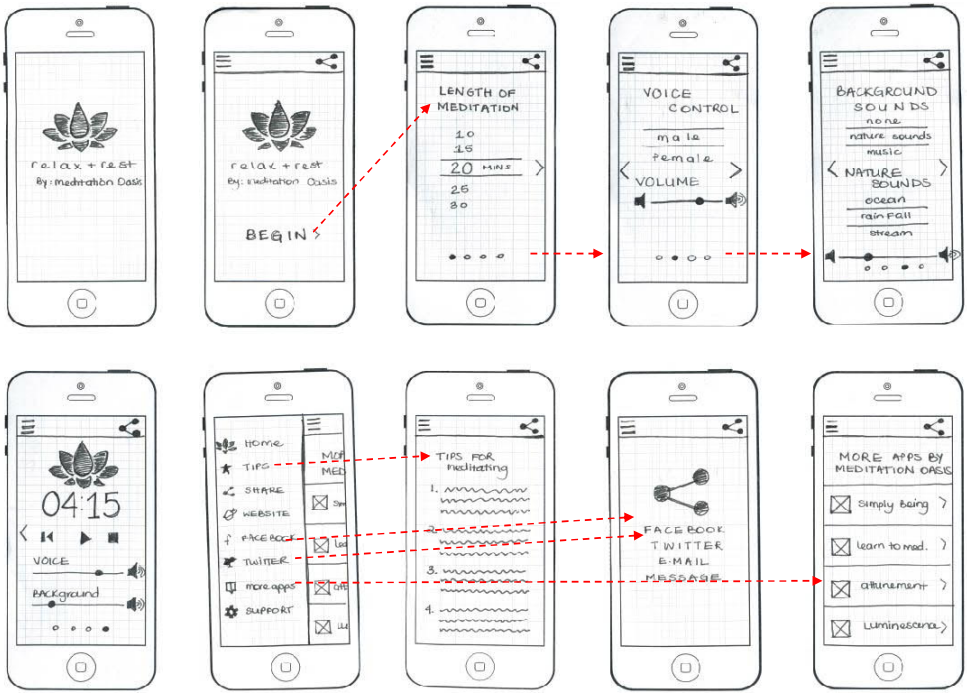
Paper prototyping (FI: paperiproto) is a method when customer or end-user is shown GUI pictures one after another, and according to their ”actions” next paper is shown. That imitates user’s functionality navigation in the program (what can be done and how).

https://i.pinimg.com/originals/f6/22/65/f622650597b13ae85a26f98f5595bfbb.png

Paper prototype is quick, easy, and powerful



https://tmecley.files.wordpress.com/2014/03/paper-prototype.jpg



Safety

a few more requirement's attributes

Safety = e.g. safe to use, for users (sw does nothing it should not do, warns user).

Security = e.g. information security, cyber security (architectural issues, user rights).

Usability = e.g. easy to use, user understands how to do some task.

Requirements should be measurable (not miserable).

Software Safety

Primary directive:

Software must do what it is intended to do.

Software should do nothing it is not meant to do.

Application areas e.g.

- machinery
- healthcare
- aerospace
- road vehicles
- military.

Functional safety of electrical/electronic/programmable electronic safety-related systems

IEC 61508 standards family

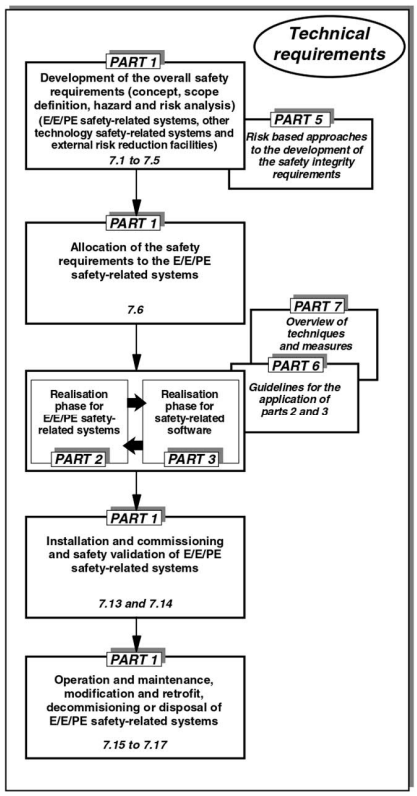


Figure 1 Requirements map for parts 1 to 7 of IEC 61508

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements

(IEC 61508-3:2010)

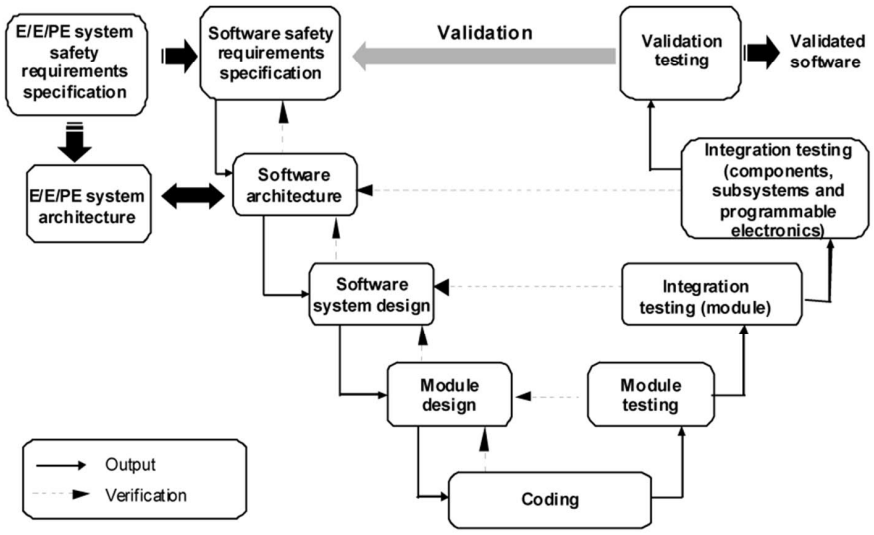
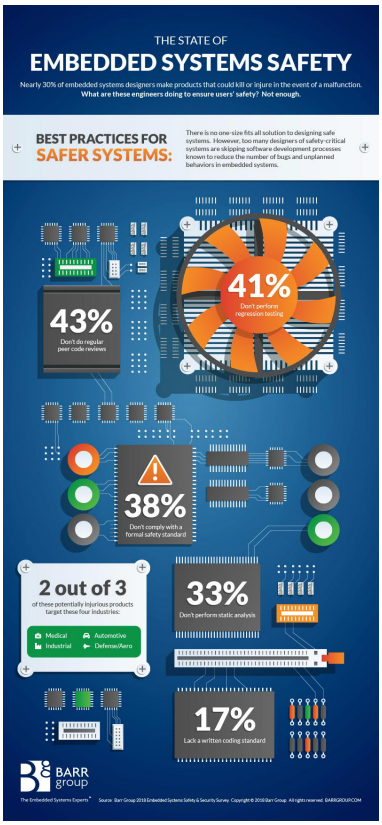


Figure 6 Software systematic capability and the development lifecycle (the V-model)

Of the industry-recommended design practices highlighted in the survey, the following statistics are of greatest concern:

43% don't do regular code reviews
41% don't perform regression testing
38% don't comply with any safety standard
33% don't use a static analysis tool
17% don't follow any coding standard.



Customer's responsibilities

Developer's and customer's responsibilities

Developer's responsibilities are e.g.

- take all aspects into consideration; user groups, stakeholders,…
- make demos/prototypes often to customer
- in case of major changes, request updates to project plan (schedule, budget,…).

Customer's responsibilities are e.g.

- be committed, not just involved
- comment requirements and GUI to developer; just "OK" is not a good answer
- be loyal to fellow workres/users, not just want "personal" features to sw
- in case of major changes, request updates to project plan (schedule, budget,…).

Tampereen yliopisto
Tampere University

Requirements management

Tampereen yliopisto
Tampere University

requirements management

3.3442

requirements management

1. activities that ensure requirements are identified, documented, maintained, communicated and traced throughout the life cycle of a system, product, or service [ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering, 4.1.20]

2. provision of storing and editing capabilities, tracking history of edition, versioning, author identification, change management, time stamping, user notification for content changes, security rights control [ISO/IEC TR 24766:2009 Information technology — Systems and software engineering — Guide for requirements engineering tool capabilities, 3.3] cf. software requirements management

[ISO/IEC/IEEE 24765:2017]

requirements traceability

3.3449 requirements traceability

1. identification and documentation of the derivation path (upward) and allocation/ flow-down path (downward) of requirements in the requirements hierarchy

2. discernible association between a requirement and related requirements, implementations, and verifications

3. traceabilities in domain and application requirements respectively and those between them [ISO/IEC 26551:2016 Software and systems engineering — Tools and methods for product line requirements engineering, 3.18]

3.3450 requirements traceability matrix (RTM)

1. table that links requirements to their origin and traces them throughout the project life cycle

2. a grid that links product requirements from their origin to the deliverables that satisfy them [A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition]

[ISO/IEC/IEEE 24765:2017]

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

Further reading…

Remember ”Additional material” at Moodle
(no exam questions about ”Additional material”)

- there is material also about requirements

Tampereen yliopisto
Tampere University

Highlights - What to remember

- requirements (= WHAT the system is supposed to do) are crucial
- perhaps would be good to know also: WHY customer needs those requirements
- if there would be mind-reading, we wouldn’t need requirements specification
- end-users might know best what they need
- several stakeholders may have different goals and needs
- requirements may be, and usually are somehow, contradicting (trade-offs)
- beware of analysis paralysis – do it agile way, start now and show protos to customer
- during a project, requirements (customer’s mind or external interactions) do change
- also outside world (e.g. business) may change during project’s time
- requirements should be measurable (not like ”easy to use”)
- safety first – also with modern software and its thousands of uses
- customer’s commitment is needed !

Now the additional L3 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L3 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L3 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Feasibility study / preliminary analysis

ISO/IEC/IEEE 24765:2017

3.1581

feasibility study

1. study to identify and analyze a problem and its potential solutions in order to determine their viability, costs, and benefits.

3.3431

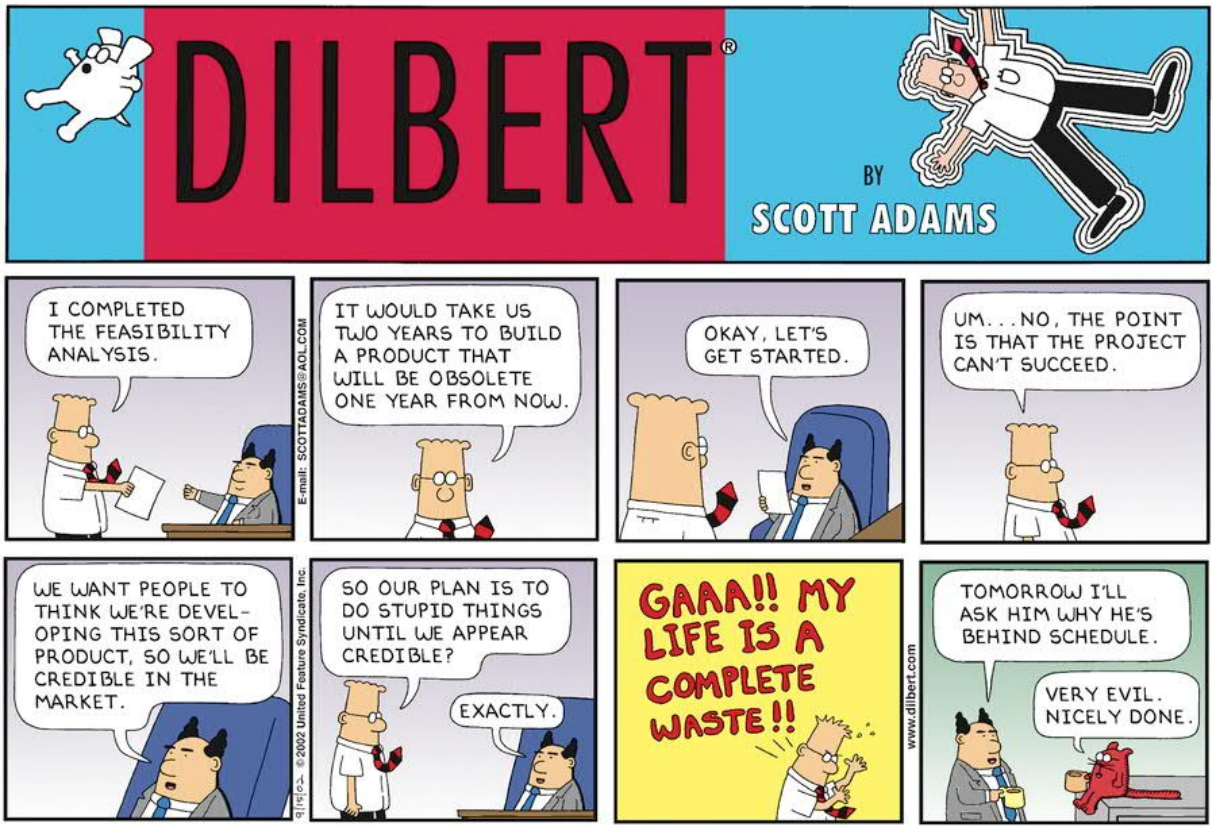
requirement

1. statement that translates or expresses a need and its associated constraints and conditions

2. condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents

3. provision that contains criteria to be fulfilled

4. a condition or capability that must be present in a product, service, or result to satisfy a contract or other formally imposed specification.



Requirements elicitation

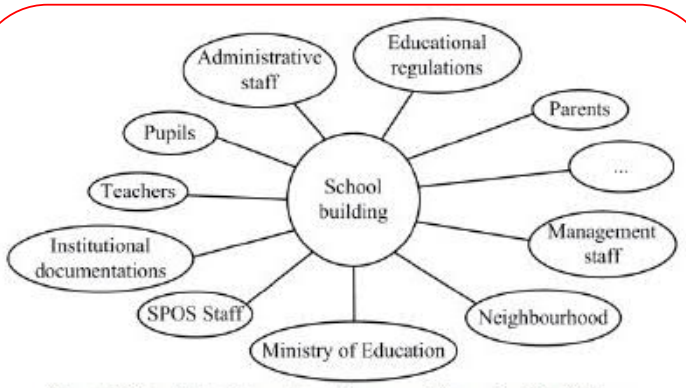


Fig. 6. Part of the interaction diagram of the school building.

TABLE I
SOURCES WITHHELD AND TYPES

Sources		Type		
Potential	Withheld	Users	Clients	Standards
Pupils		X		
Teachers	X	X		
SPOS staff	X	X		
Administrative staff	X	X		
Management staff	X	X		
Ministry of Education			X	
Educational laws and regulations	X			X
Institutional documentation	X			X

TABLE II
TECHNIQUES WITHHELD FOR ELICITATION PROCESS

Elicitation process steps	Techniques						
	Domain Analysis	Unstructured Interview	Observation	Brainstorming	Group Work	Task Analysis	Scenarios
Understanding the application domain	X	X	X			X	X
Identifying the sources	X			X	X		
Analyzing the stakeholders		X					
Selecting the techniques	X						
Eliciting the requirements	X	X	X	X	X		

[Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010]

Tampereen yliopisto
Tampere University

Chapter 1: Introduction

Chapter 2: Business Analysis Key Concepts

Chapter 3: Business Analysis Planning and Monitoring

Chapter 4: Elicitation and Collaboration

Chapter 5: Requirements Life Cycle Management

Chapter 6: Strategy Analysis

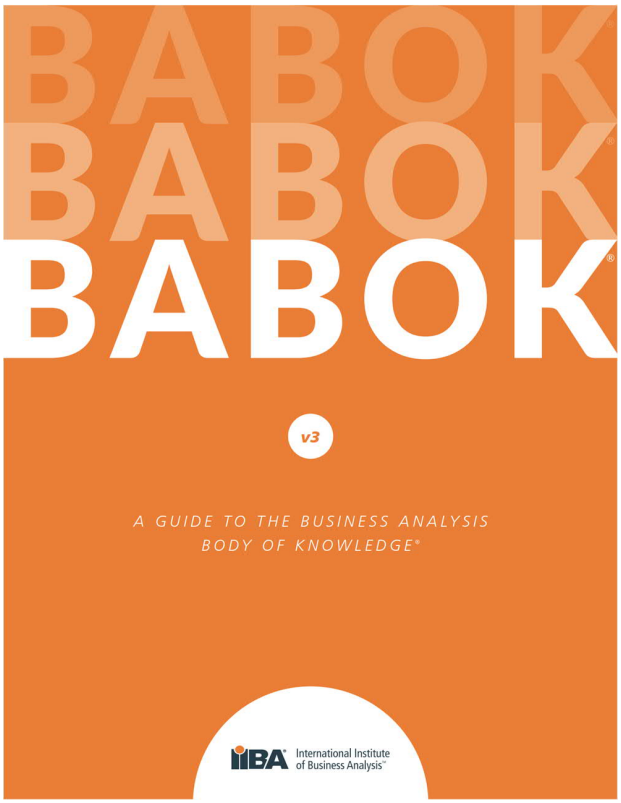
Chapter 7: Requirements Analysis and Design Definition

Chapter 8: Solution Evaluation

Chapter 9: Underlying Competencies

Chapter 10: Techniques

Chapter 11: Perspectives.



Tampereen yliopisto
Tampere University

CHAPTER 1: PREFACE AND INTRODUCTION

CHAPTER 2: ENTERPRISE ANALYSIS

CHAPTER 3: REQUIREMENTS PLANNING AND MANAGEMENT

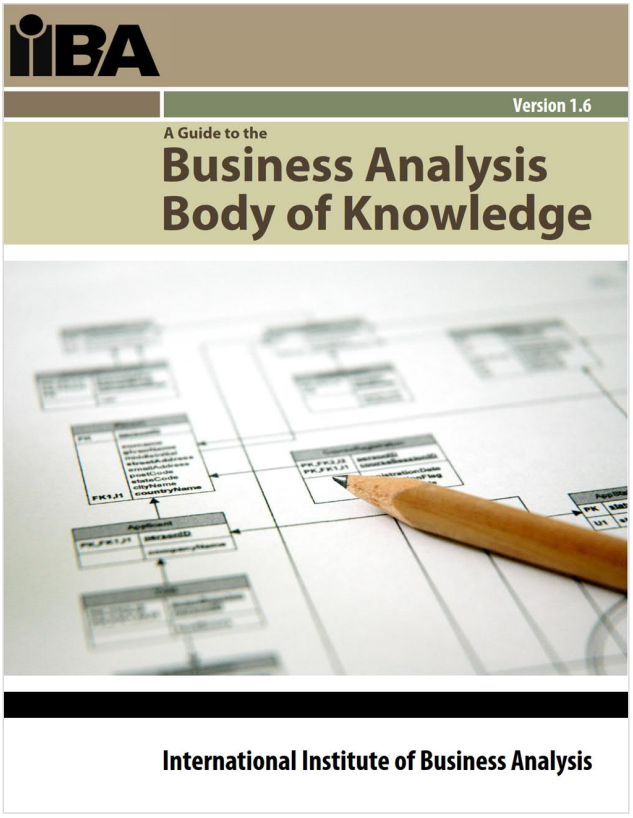
CHAPTER 4: REQUIREMENTS ELICITATION

CHAPTER 5: REQUIREMENTS ANALYSIS AND DOCUMENTATION

CHAPTER 6: REQUIREMENTS COMMUNICATION

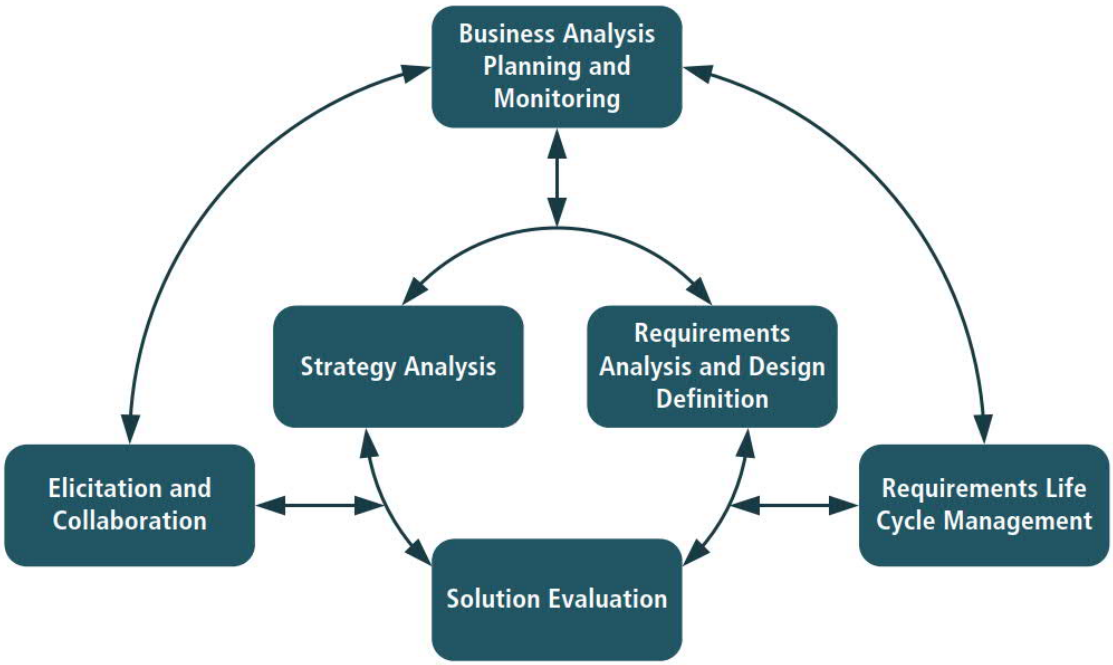
CHAPTER 7: SOLUTION ASSESSMENT AND VALIDATION

CHAPTER 8: BA FUNDAMENTALS.



Tampereen yliopisto
Tampere University

Figure 1.4.1: Relationships Between Knowledge Areas



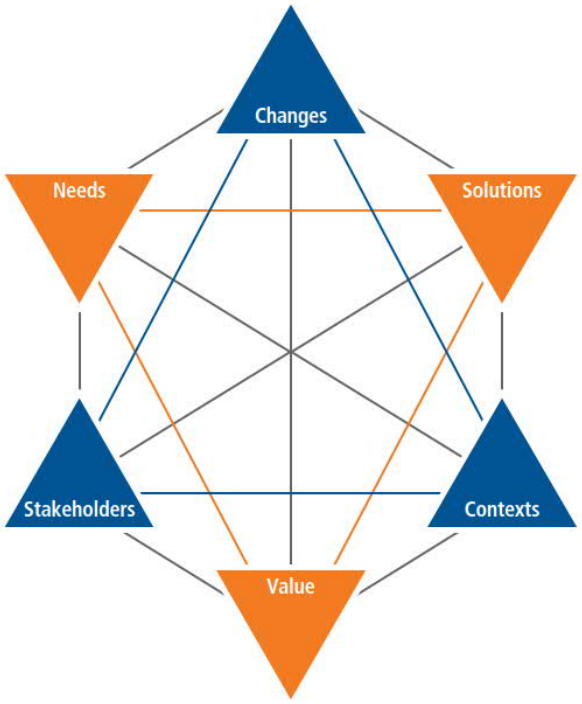
[BABOK v3, 2015]

Tampereen yliopisto
Tampere University

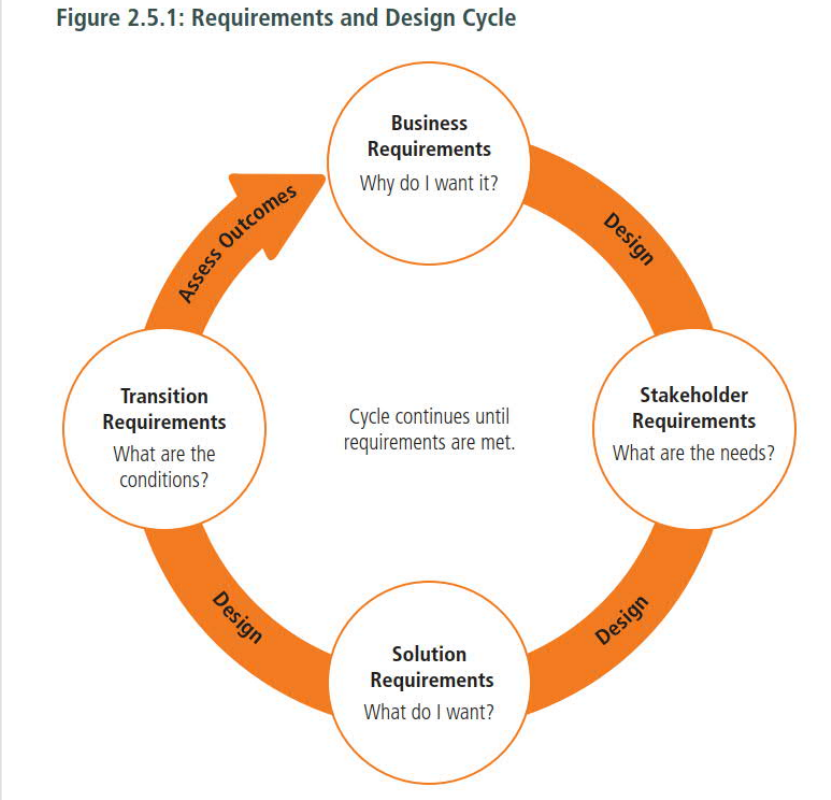
Business Analysis Core Concept Model™ (BACCM™):

defines a conceptual framework for the business analysis profession.

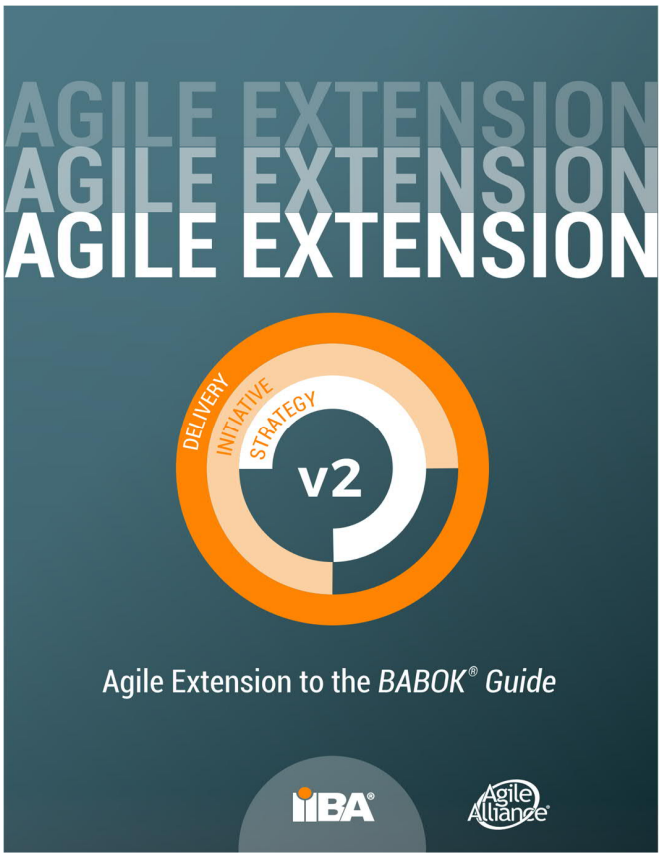
Figure 2.1.1: The BACCM



[BABOK v3, 2015]



[BABOK v3, 2015]



Stakeholders

[ISO 21500:2012]

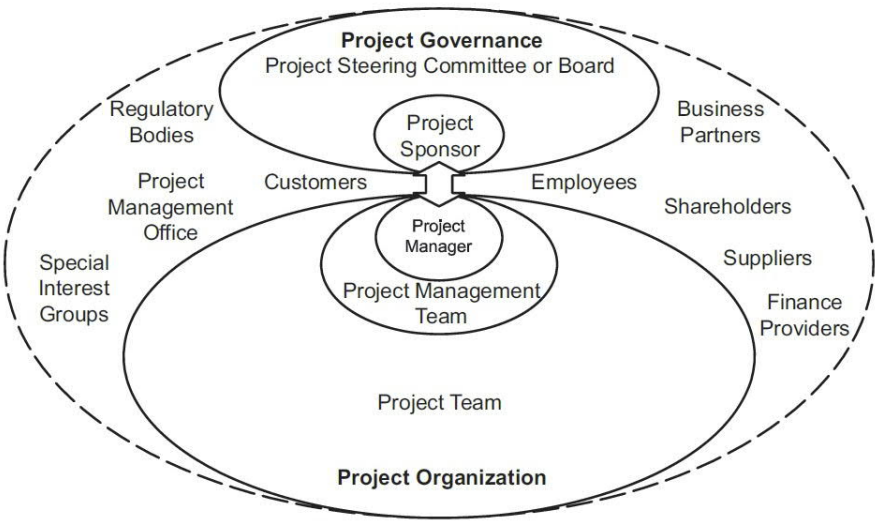
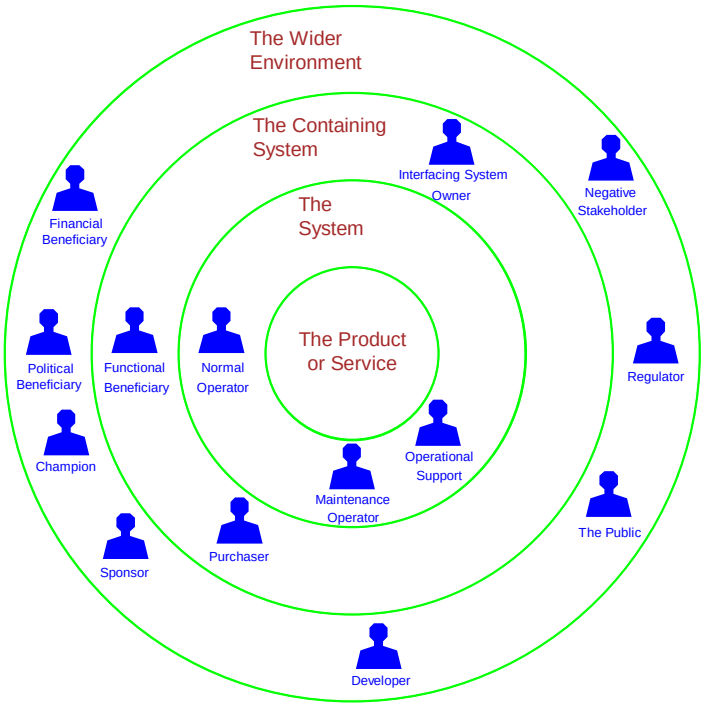


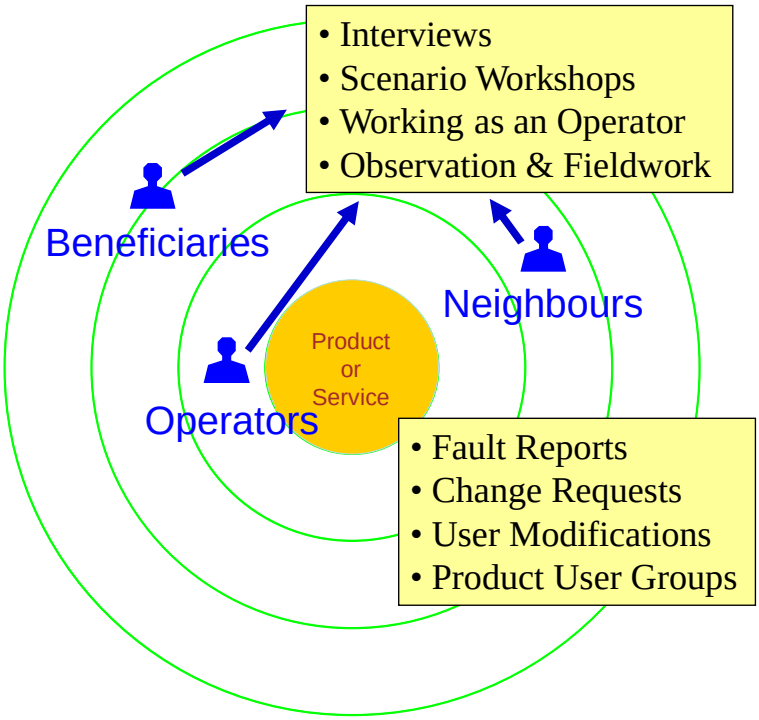
Figure 4 — Project stakeholders

Stakeholders in a Typical System



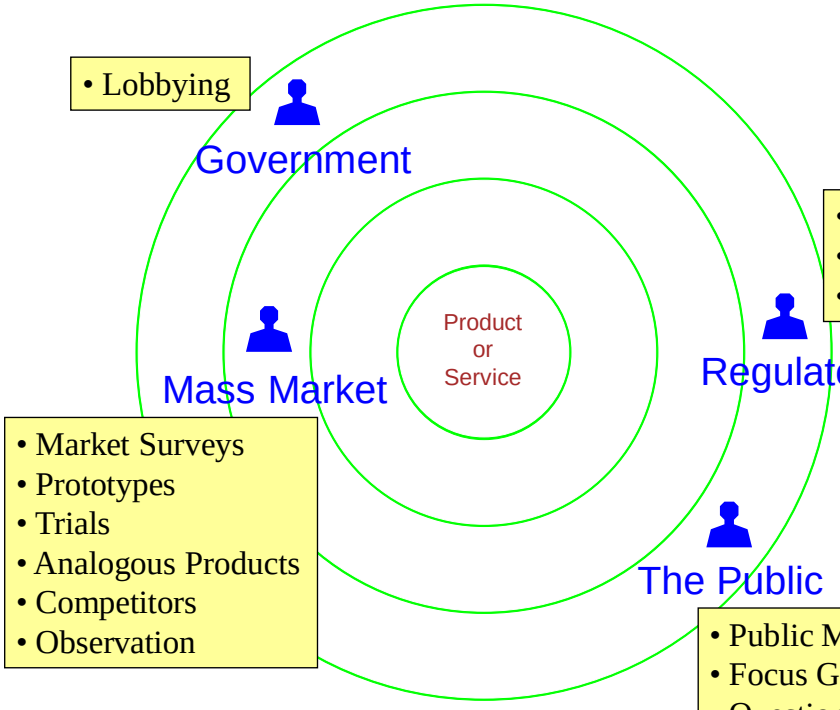
[scenarioplus.org.uk]

Discovery Contexts and Sources

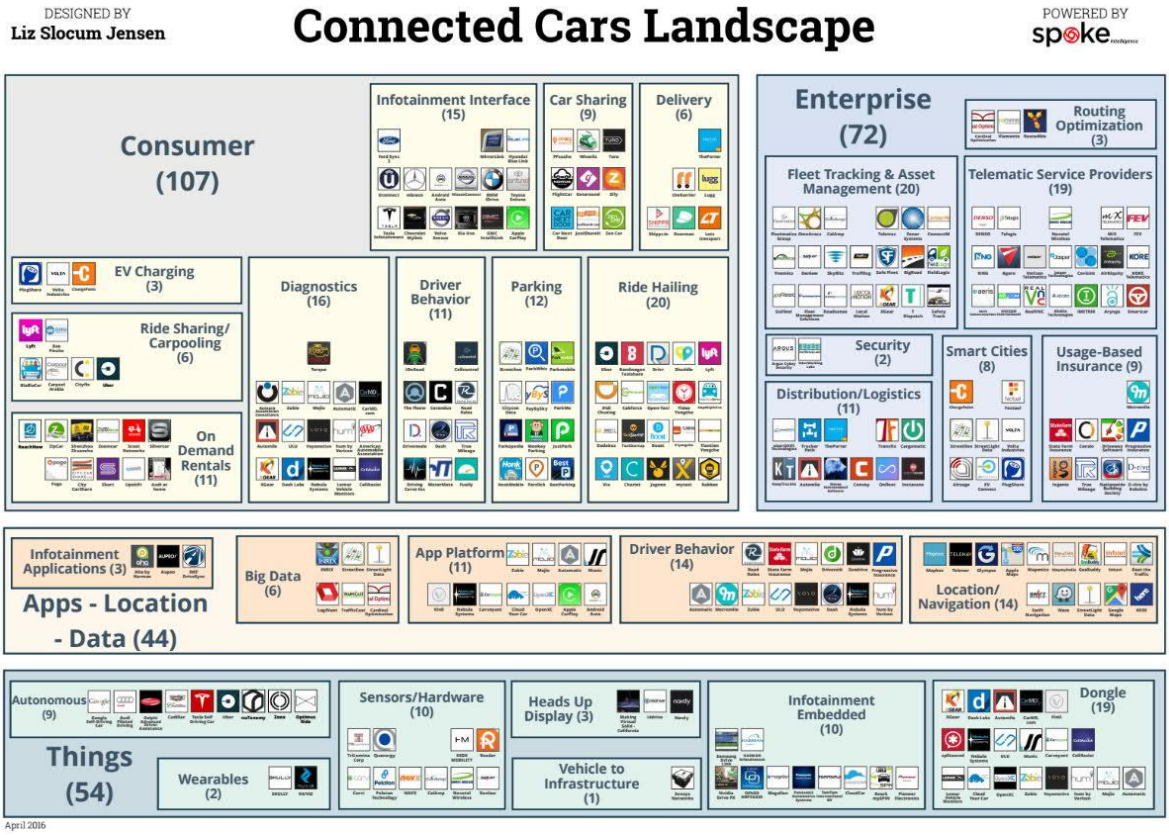


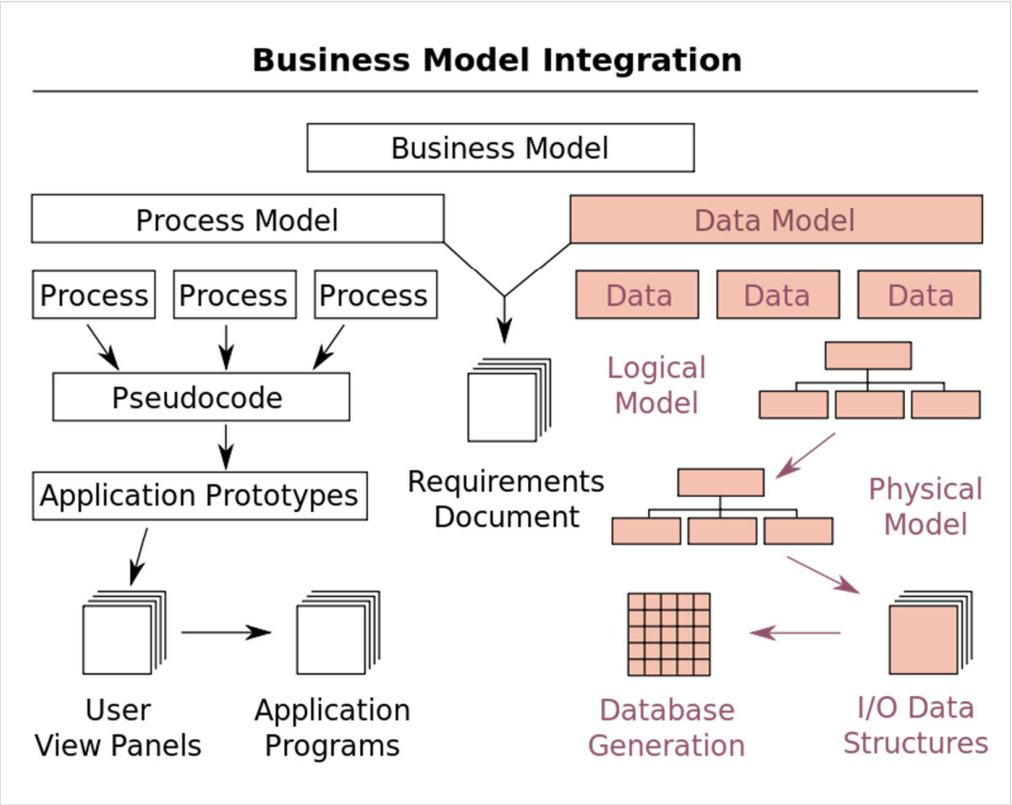
[scenarioplus.org.uk]

Discovering from Non-Operational Roles



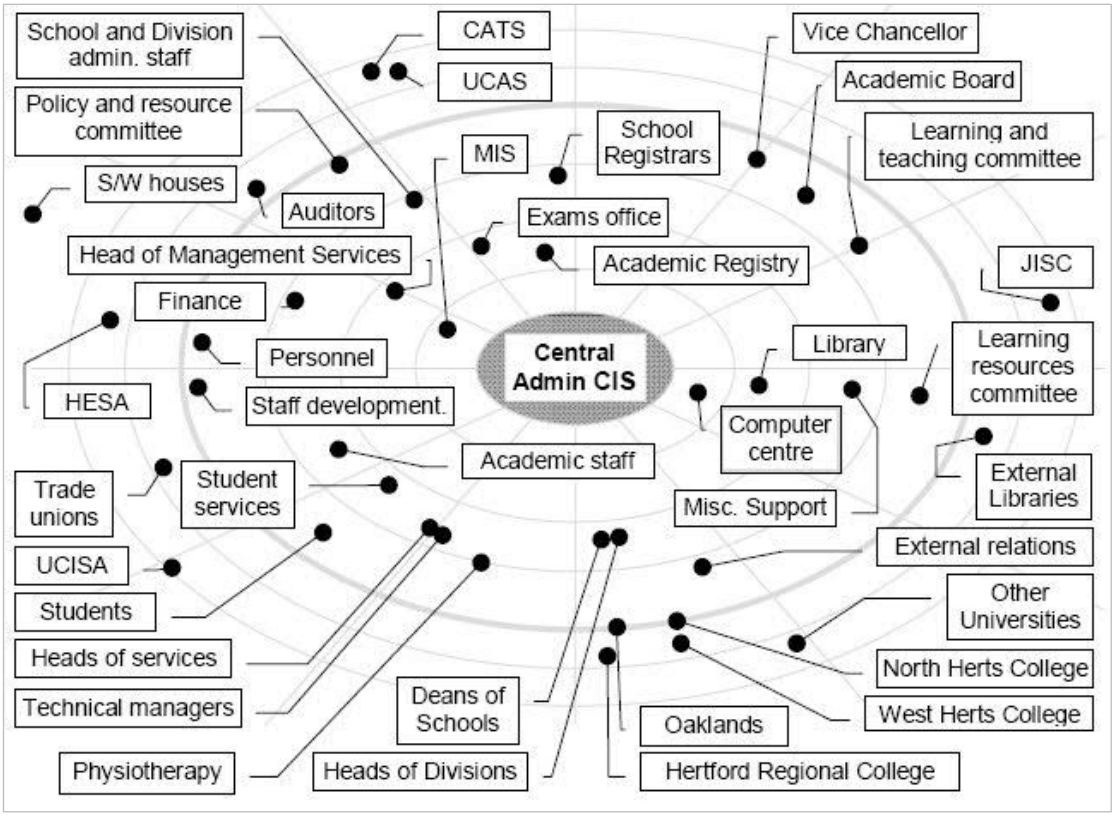
[scenarioplus.org.uk]





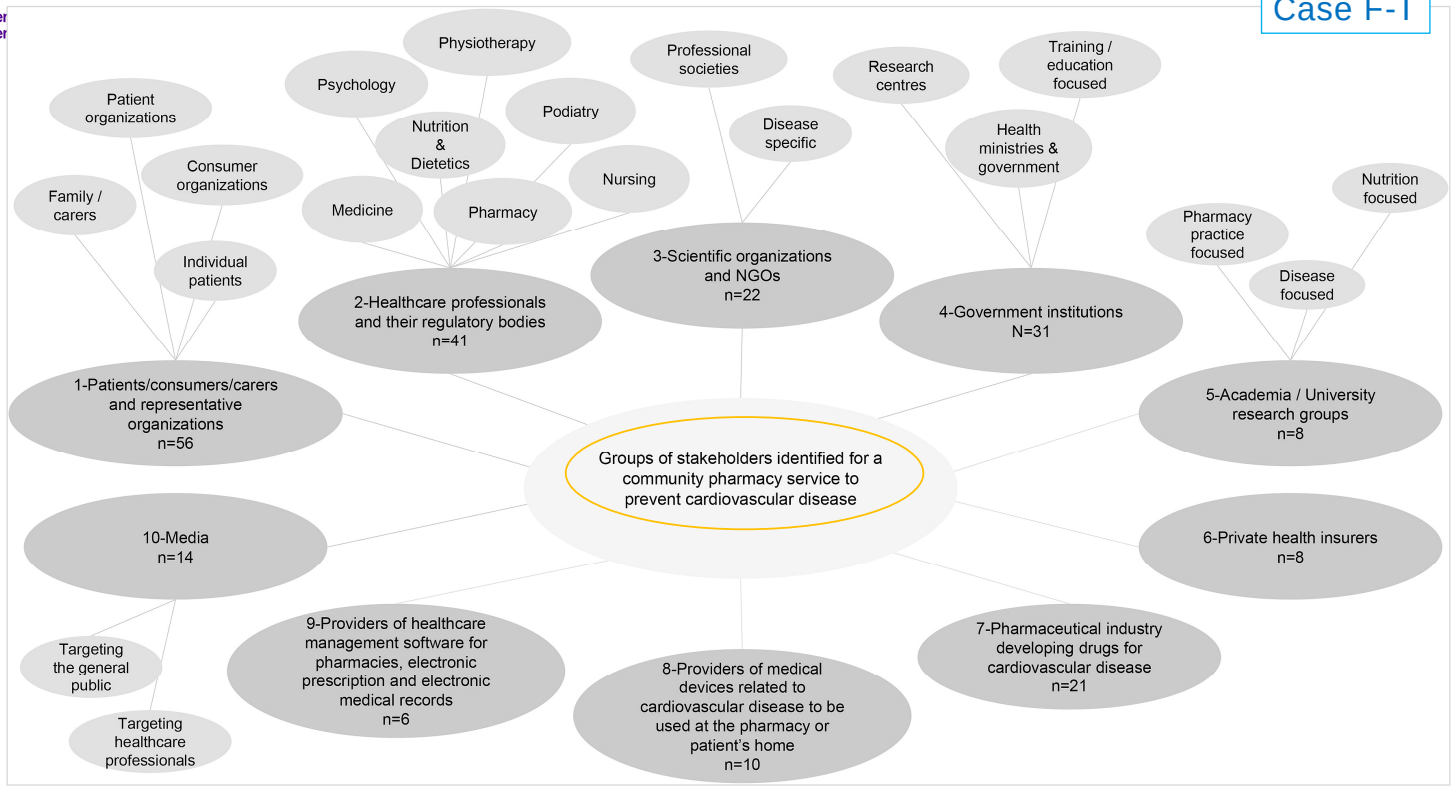
[Paul R. Smith. Redrawn by Marcel Douwe Dekker]

University example



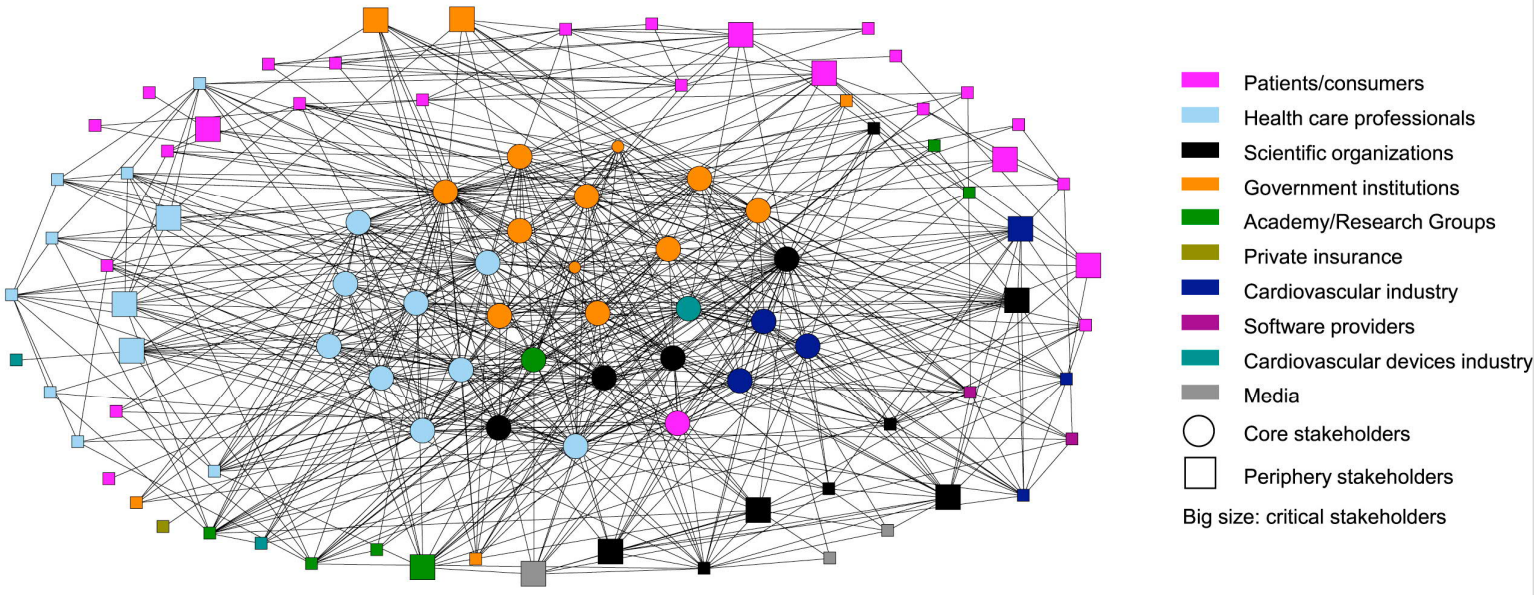
[scenarioplus.org.uk]

Case F-T



Tampereen yliopisto
Tampere University

Case F-T: Collaboration network about stakeholders.



[L. Franco-Trigo et al.: Collaborative health service planning: A stakeholder analysis with social network analysis to develop a community pharmacy service, 2019]

Requirements

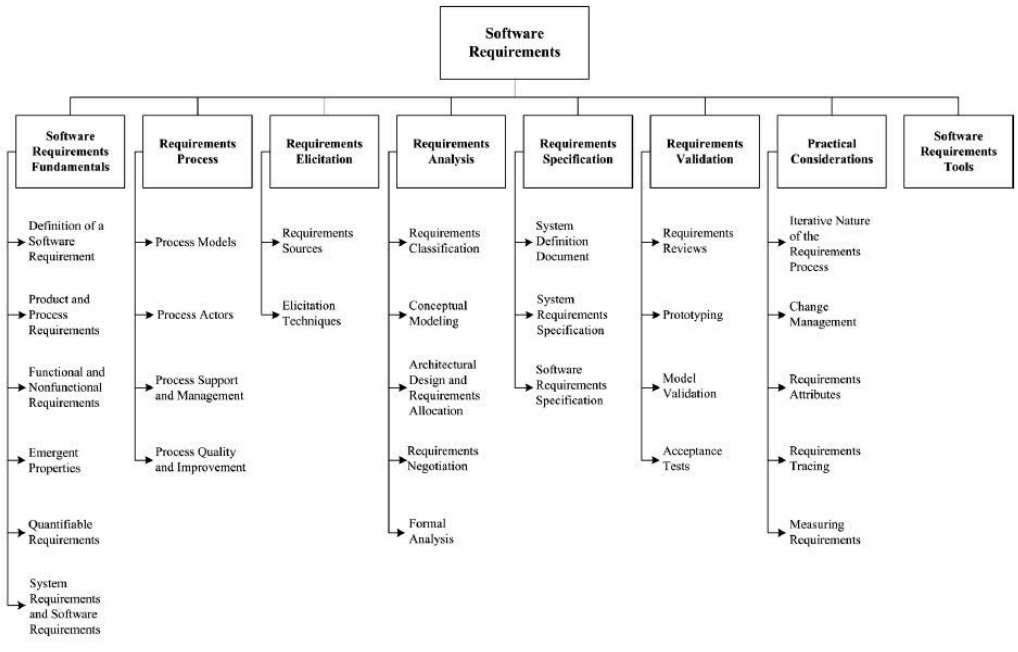


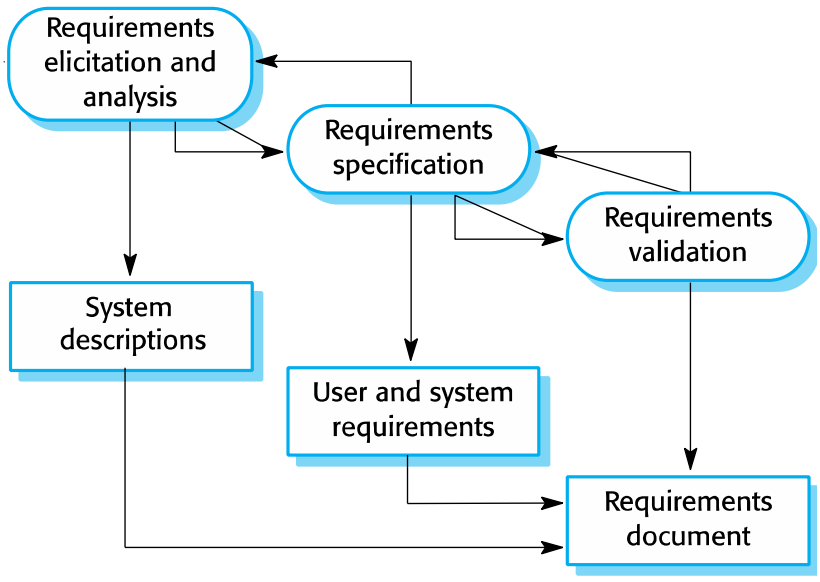
Figure 1.1. Breakdown of Topics for the Software Requirements KA

Weeding out requirements from stakeholders… remember to be polite.

The requirements engineering process

[Sommerville 2014]

Requirements engineering is developing a software specification.



Functional and non-functional requirements

[Sommerville 2014]

✧ Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- May state what the system should not do.

✧ Non-functional requirements

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Often apply to the system as a whole rather than individual features or services.

✧ Domain requirements

- Constraints on the system from the domain of operation.

Tampereen yliopisto
Tampere University

[Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010]

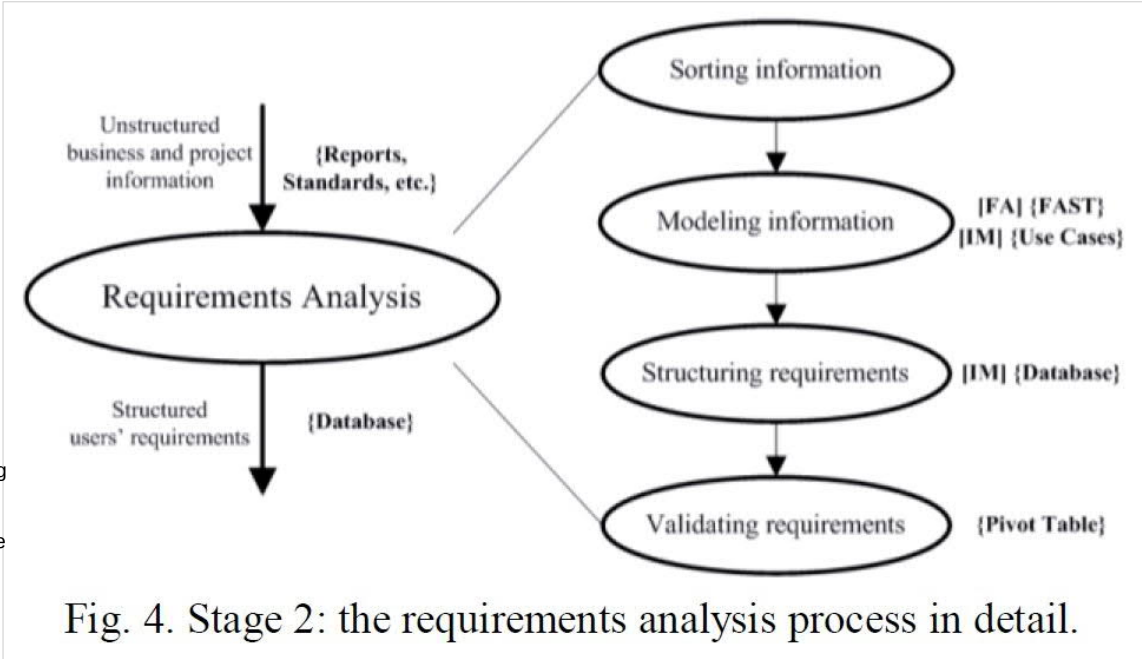
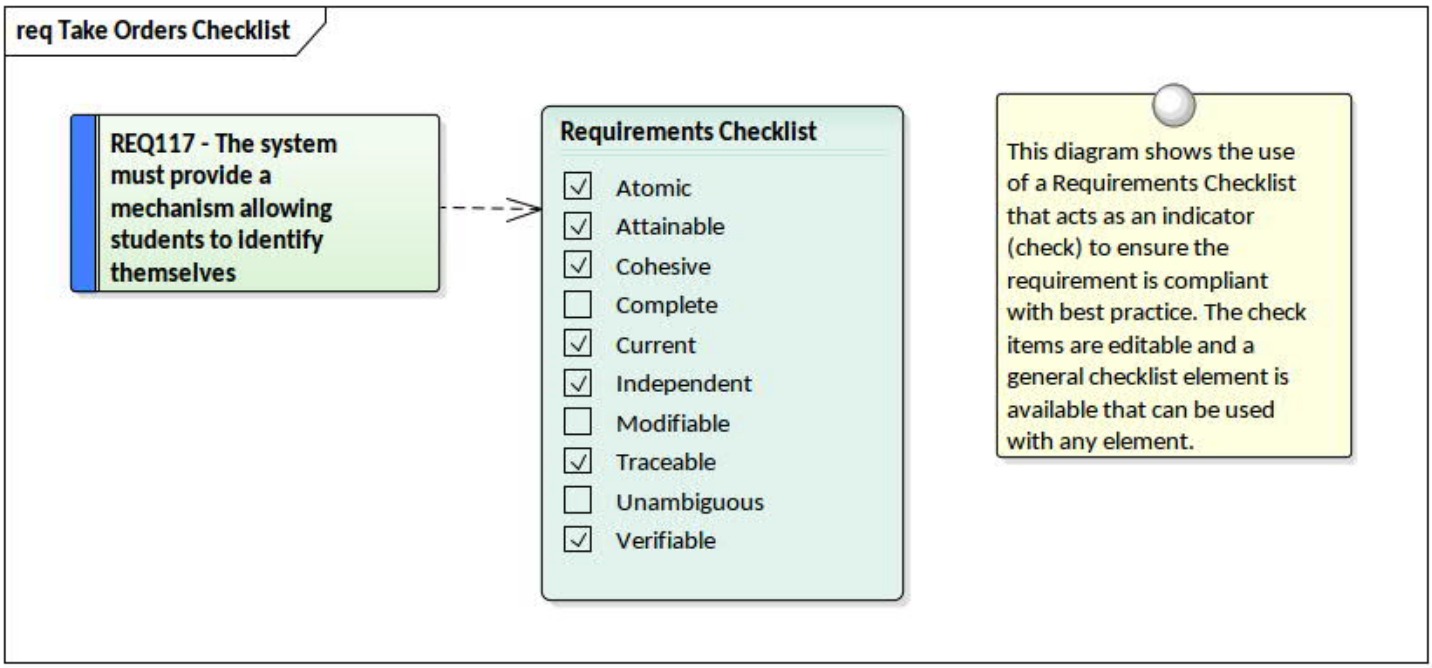


Fig. 4. Stage 2: the requirements analysis process in detail.



[sparxsystems.com/resources/gallery/diagrams/]

Kahoot query (quiz);

What is a good DI (Master) like ?

- every graduated DI is good !
- the DI who has grade 4 or 5
- the DI who graduated within five years
- the DI who has full-time work at graduation.

FI: DI = diplomi-insinööri (Master of Science, tech/eng).

Just do what is required, don't mess

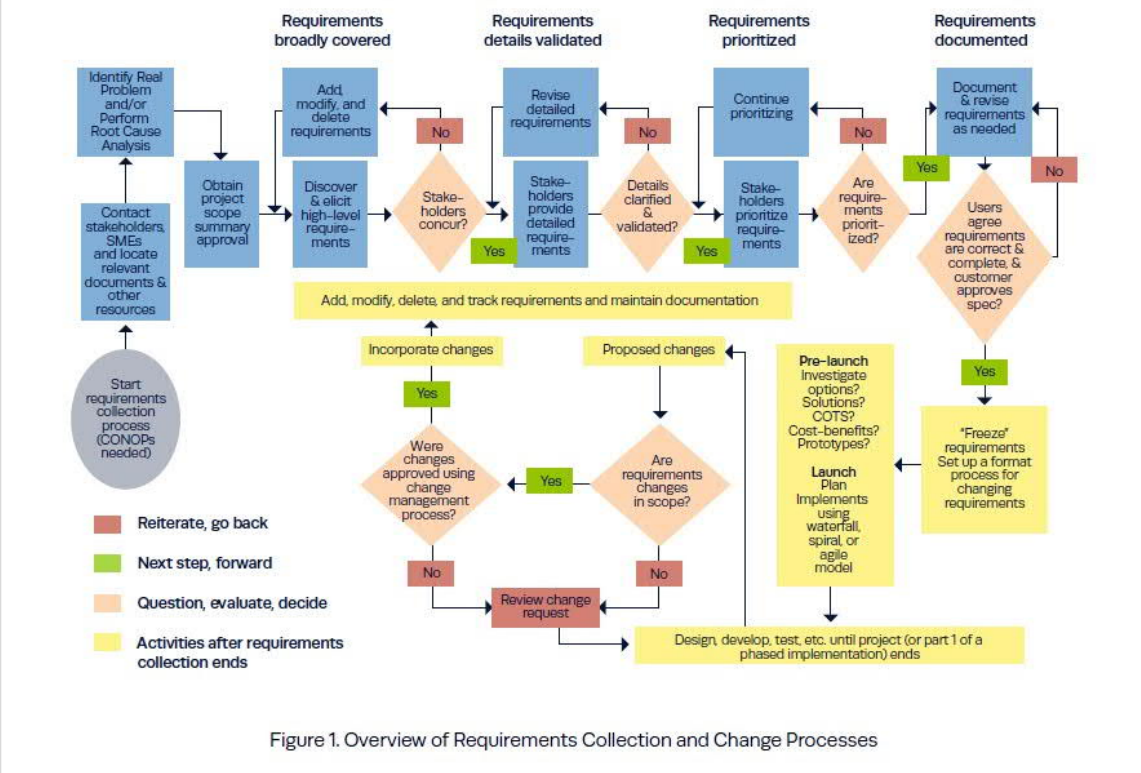


Figure 1. Overview of Requirements Collection and Change Processes

CONOPS = concept of operations, document. [MITRE, 2014]

ISO 29148:2018 Systems and software engineering — Life cycle processes — Requirements engineering

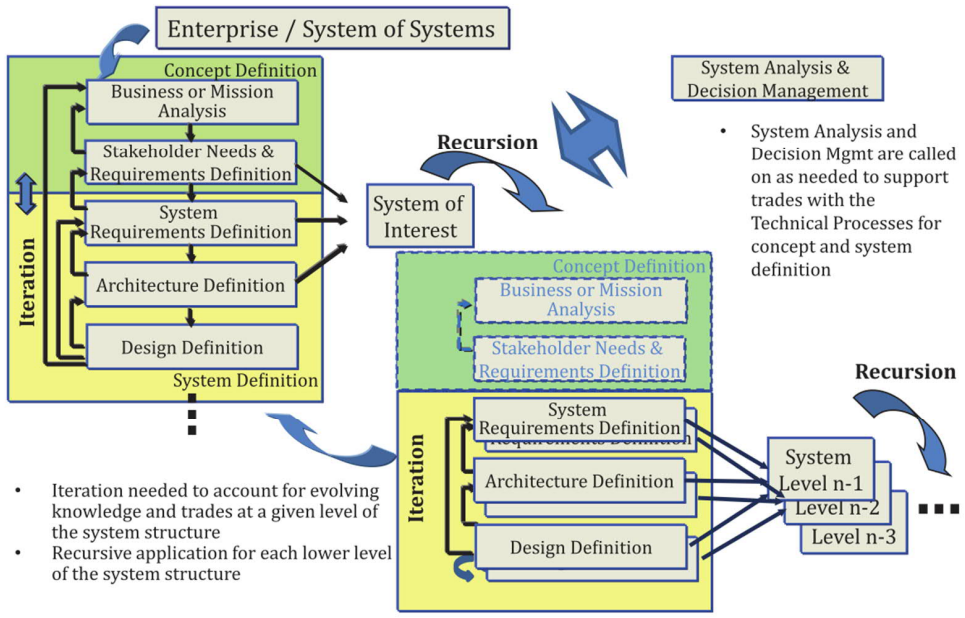


Figure 2 — Iterative and recursive application of processes

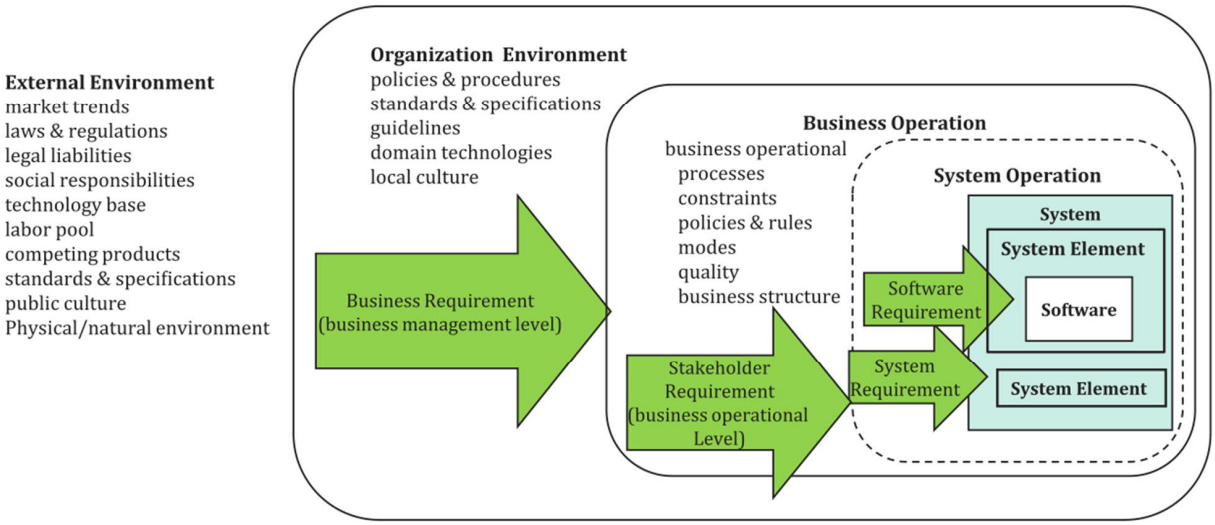


Figure 3 — Example of requirements scope in a business context

ISO/IEC/IEEE 29148:2018(E)

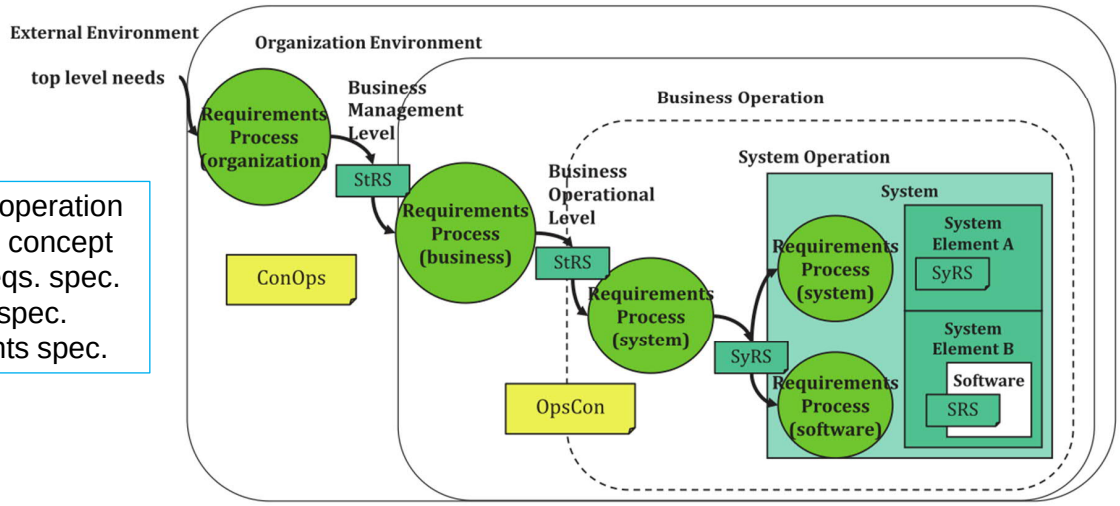


Figure 4 — Example of the relationships between requirements processes and specifications

ConOps = concept of operation
OpsCon = operational concept
StRS = stakeholder reqs. spec.
SyRS = system reqs. spec.
SRS = Sw requirements spec.

Christopher Lindquist: Software Requirements Mess

Hugh Cumming had his work cut out for him. The gap between what his not-yet-implemented call center management application at a large European company could do and the requirements list created by 40 eager business-side stakeholders now filled 3,000 pages and threatened to delay an already overdue call center consolidation effort another four to five years. "My first instinct was that the project had absolutely no chance of success," says Cumming, currently CIO for ADP Employer Services Canada.

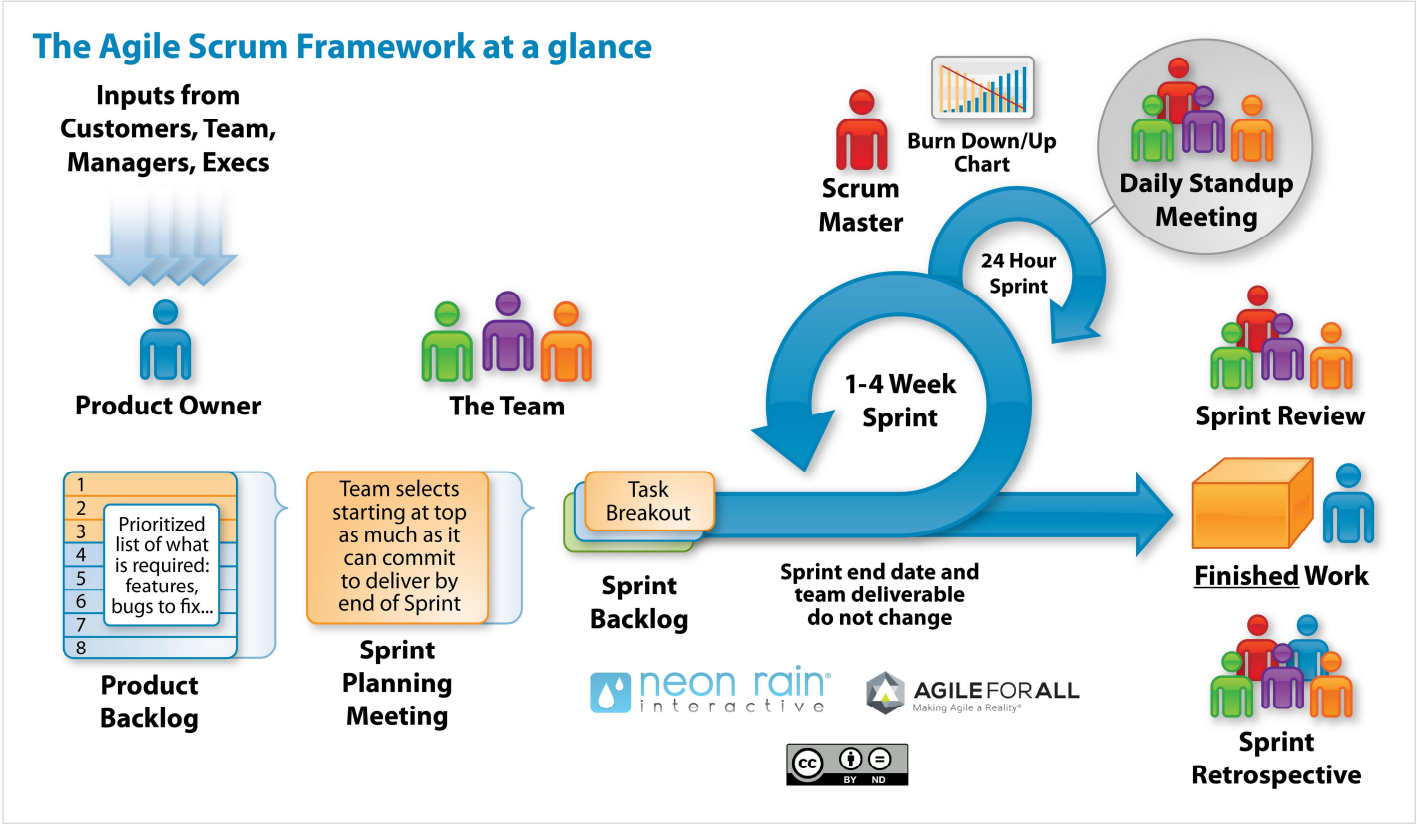
Requirements, as every CIO knows, are a problem, but CIOs may not be aware of just how catastrophic the problem has become. **Analysts report that as many as 71 percent of software projects that fail do so because of poor requirements management**, making it the single biggest reason for project failure—bigger than bad technology, missed deadlines or change management fiascoes. Though CIOs are rarely directly responsible for requirements management, they are accountable for poor outcomes, which, when requirements go bad, can include: project delays, software that doesn't do what it's supposed to and, worst of all, software that may not work correctly when rolled out, putting the business—and the CIO's job—at risk.

Mishandled requirements can torpedo a project at any time, from inception to delivery. **Start down the wrong road and you arrive at the wrong destination. And even if you're heading in the right direction, making fumbling changes midstream can be almost as deadly.**

CIO = Chief Information Officer (FI: tietohallintojohtaja)

What to write about a requirement

- date (of origin)
- author/writer
- source (customer and why…)
- type (change, addition, correction)
- description
- dependence to/from other requirements (if any)
- need (required, optional = nice to have, extra = additional)
- propability of change (will not, maybe, is likely) during requirements process.



Government design principles

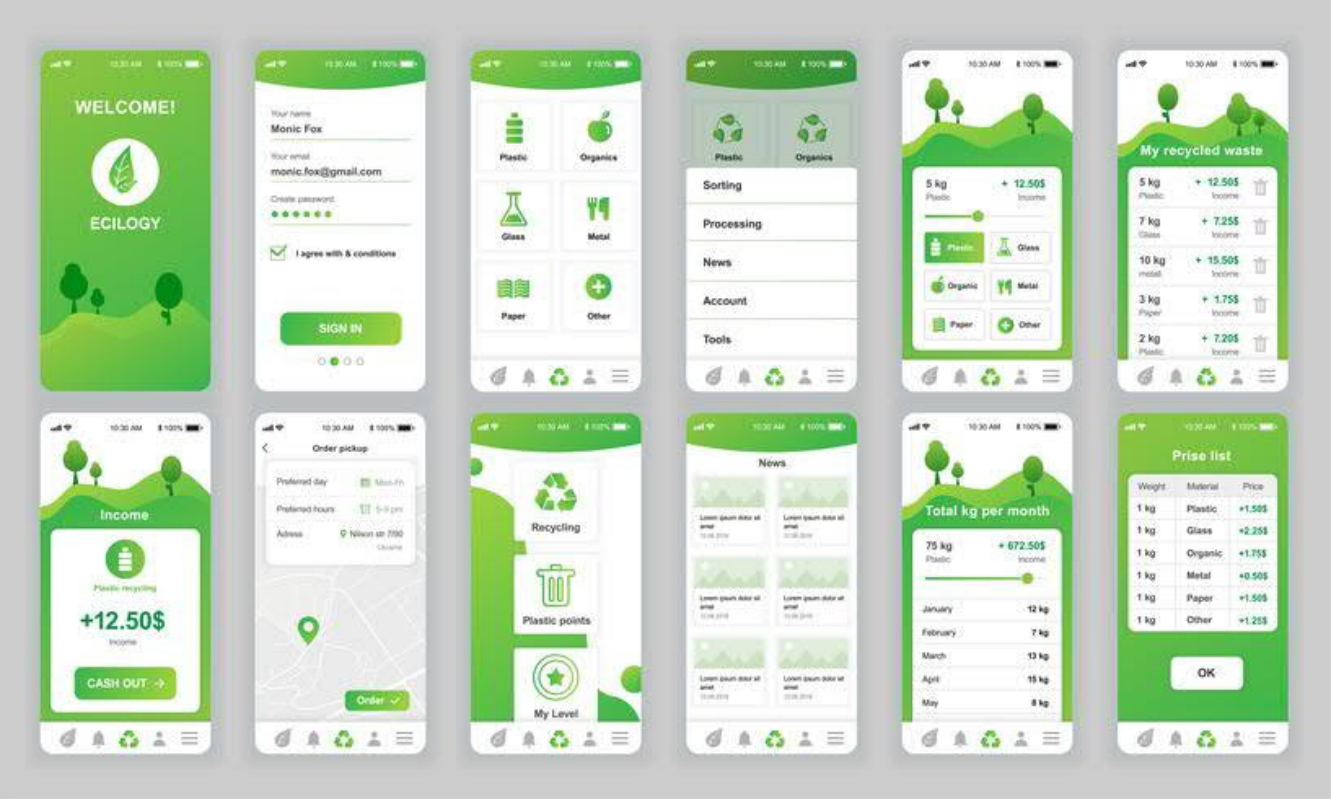
The UK government's design principles and examples of how they've been used.
Published 3 April 2012
From: Government Digital Service

Contents

1. Start with user needs
2. Do less
3. Design with data
4. Do the hard work to make it simple
5. Iterate. Then iterate again
6. This is for everyone
7. Understand context
8. Build digital services, not websites
9. Be consistent, not uniform
10. Make things open: it makes things better.

https://www.gov.uk/guidance/government-design-principles

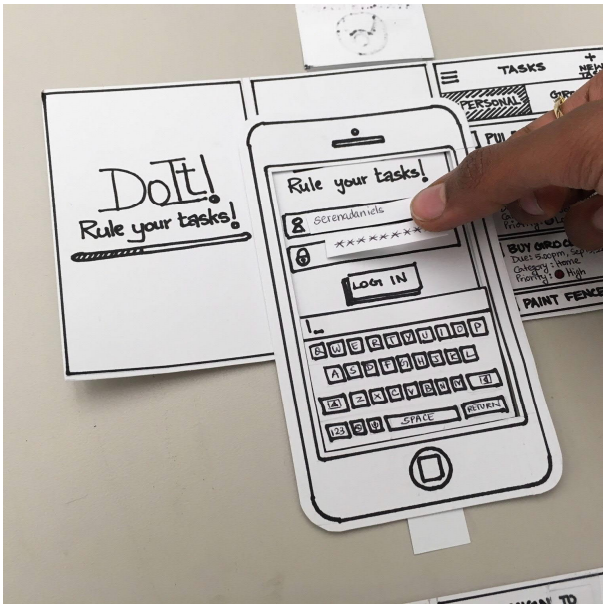
GUI wireframe



https://ergomania.eu/wp-content/uploads/33_smart-paper-prototype-1.jpg

https://martha-eierdanz.com/

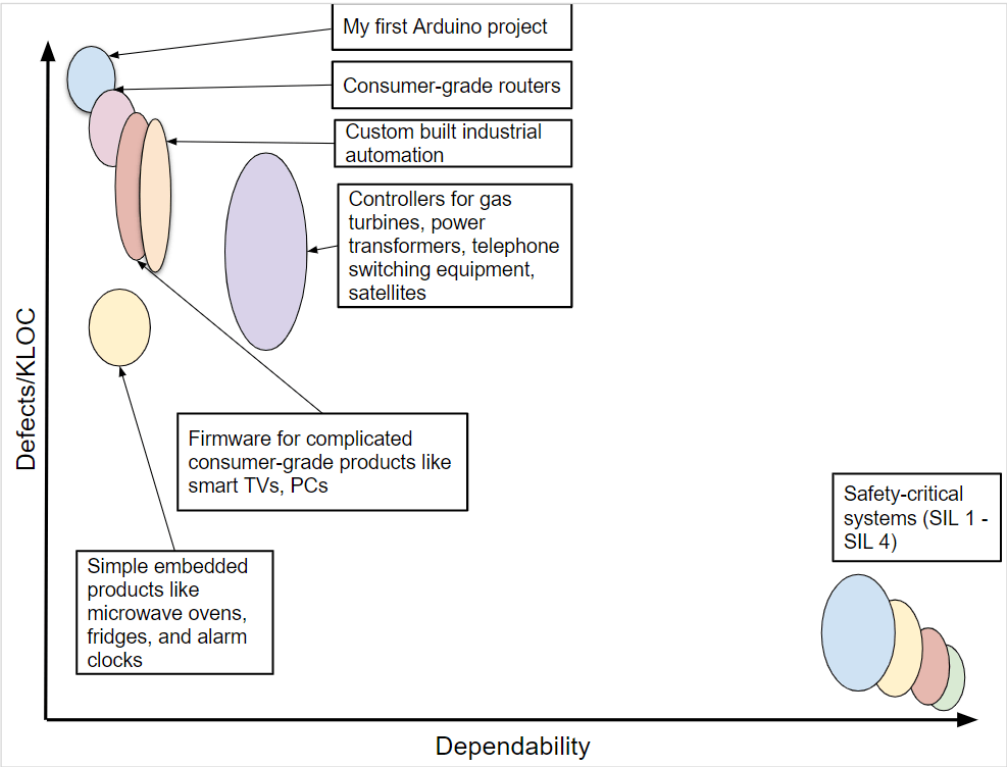
https://i.pinimg.com/

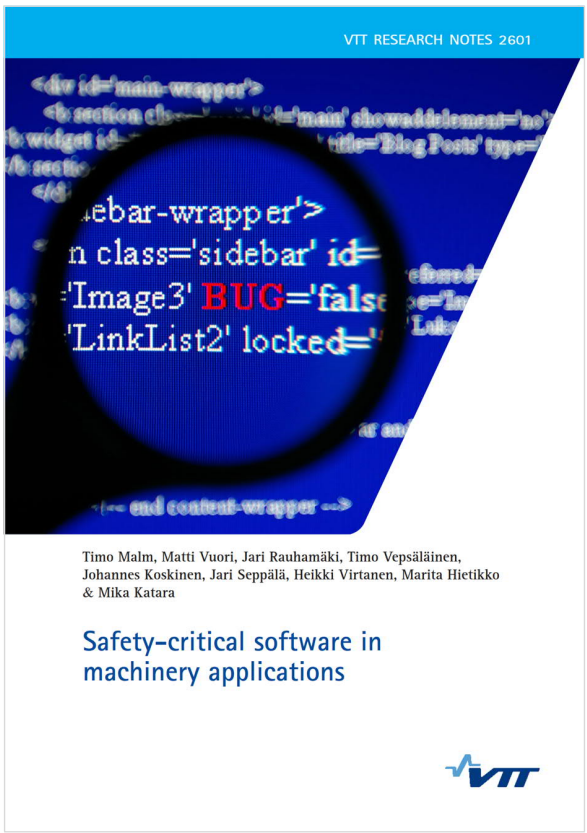


Some different kind of paper prototypes.

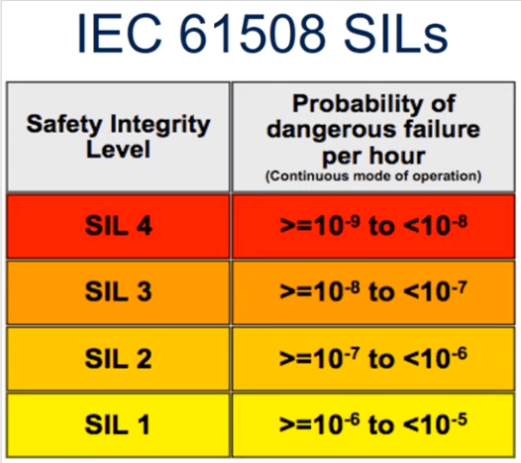
Tampereen yliopisto
Tampere University

Safety


My first Arduino project
Consumer-grade routers
Custom built industrial automation
Controllers for gas turbines, power transformers, telephone switching equipment, satellites
Firmware for complicated consumer-grade products like smart TVs, PCs
Simple embedded products like microwave ovens, fridges, and alarm clocks
Safety-critical systems (SIL 1 - SIL 4)
Defects/KLOC
Dependability


VTT RESEARCH NOTES 2601
Timo Malm, Matti Vuori, Jari Rauhamäki, Timo Vepsäläinen, Johannes Koskinen, Jari Seppälä, Heikki Virtanen, Marita Hietikko & Mika Katara
Safety-critical software in machinery applications
VTT

https://smallbusinessprogramming.com/safety-critical-software-15-things-every-developer-should-know/



IEC 61508 SILs

Safety Integrity Level	Probability of dangerous failure per hour (Continuous mode of operation)
SIL 4	$>=10^{-9}$ to $<10^{-8}$
SIL 3	$>=10^{-8}$ to $<10^{-7}$
SIL 2	$>=10^{-7}$ to $<10^{-6}$
SIL 1	$>=10^{-6}$ to $<10^{-5}$

Safety-Critical Software: 15 things every developer should know

MARCH 1, 2020 / BLAINE OSEPCHUK

Despite being all around us, safety-critical software isn't on the average developer's radar. But recent failures of safety-critical software systems have brought one of these companies and their software development practices to the attention of the public. I am, of course, referring to Boeing's two 737 Max crashes, the subsequent grounding of all 737 Max aircraft, and its failed Starliner test flight.

How could such a distinguished company get it so wrong? Weren't the safety standards and certification process for safety-critical systems supposed to prevent this kind of thing from happening? Where was the FAA when the Max was being certified? These questions raised my curiosity to the point that I decided to discover what this specialized field of software development is all about.

In this post I'm going to share what I learned about safety-critical software development and how a little knowledge of it might be useful to "normal" programmers like you and me.

Customer's responsibilities

https://medium.com/@exposit/customers-role-and-responsibilities-in-software-development-project-7101cdf9292e

Customer's responsibilities (at least partly)

- Providing project development requirements
- Communication
- Delivery Verification and Software Version Control
- Project budget control
- Risk identification and management.

Customer's role and responsibilities in software development project

Exposit Follow
Nov 16, 2018 · 3 min read

There are a lot of factors that can change the result of a software development project: professionalism of an outsourcing team, quality of collected requirements, project's budget, etc. But the quality of the developed product depends on both the implementing partner and the client. Let's see what is the client's role in a software development project.

http://www.project-management-insights.com/project-roles/customer/

Project Management Insights

enter keywords here Search

Get a FREE advice! | PMI | Roles | Project Management Software | Project Management Templates | Contact

- The Project Manager Role
- Project Team Member Role
- The Project Sponsor Role
- The Project Customer Role
- The Functional Manager Role

SOFTWARE

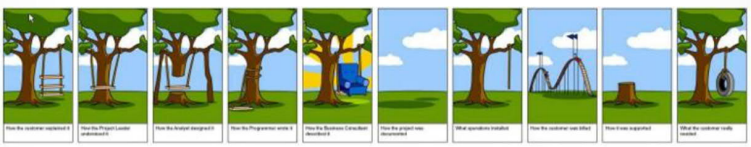
Recent Posts

PROJECT MANAGEMENT – THE BASICS OF THEORY
FLEXIBLE PROJECT MANAGEMENT TOOLS DO EXIST

Home » Project Roles » The Project Customer Role

The Project Customer Role

Anteriorly, in the 1st Part of the present series we used to totalize all the project management roles and described in more details the Project Manager role. In the 2nd Part we contemplated the Project Team Member role and the Project Sponsor role. And in this Part the **Project Customer role** will be studied in details.



Tampereen yliopisto
Tampere University

https://customerthink.com/role-of-a-customer-in-software-development-projects/

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

Requirements management

Tampereen yliopisto
Tampere University

WE1, pictures at the end

Last year 2019, when we had contact teaching and weekly exercises…

What was done in weekly exercise 1 (LEGO) in groups

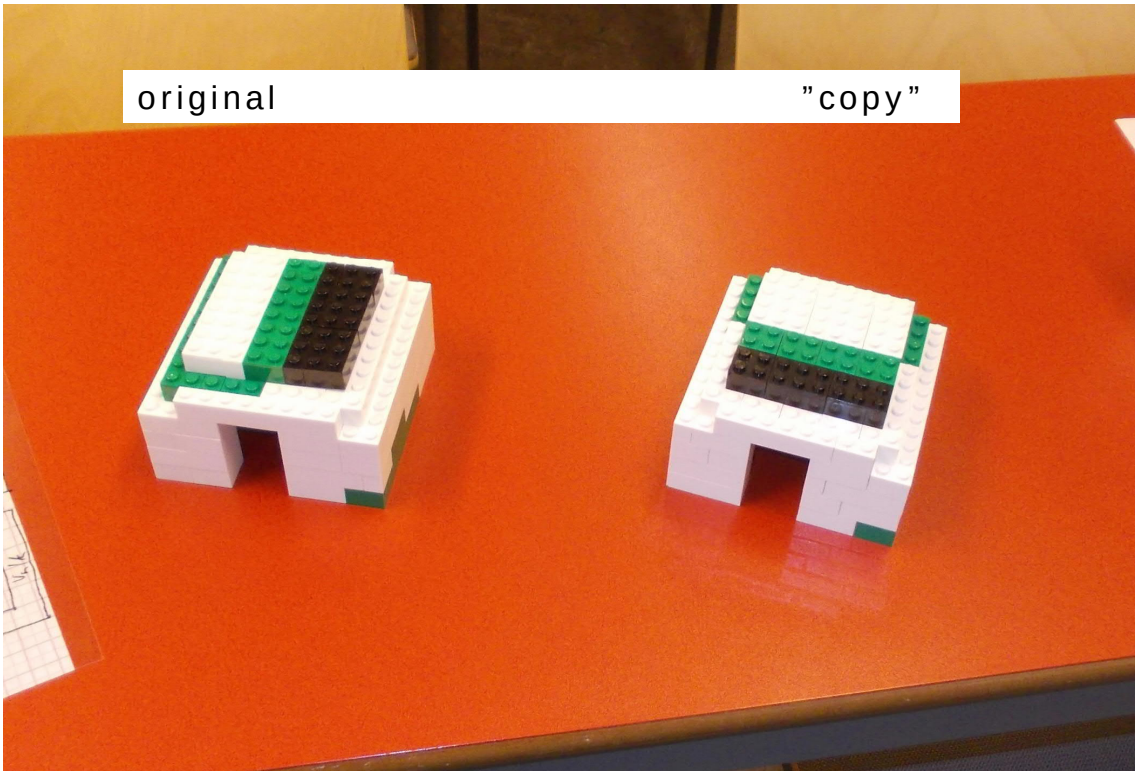
a) Do a certain device from LEGOs (there were at most five groups and five different ”devices”; house - robot - crane - space ship - cow)

b) Write specification about that device to transparency slide

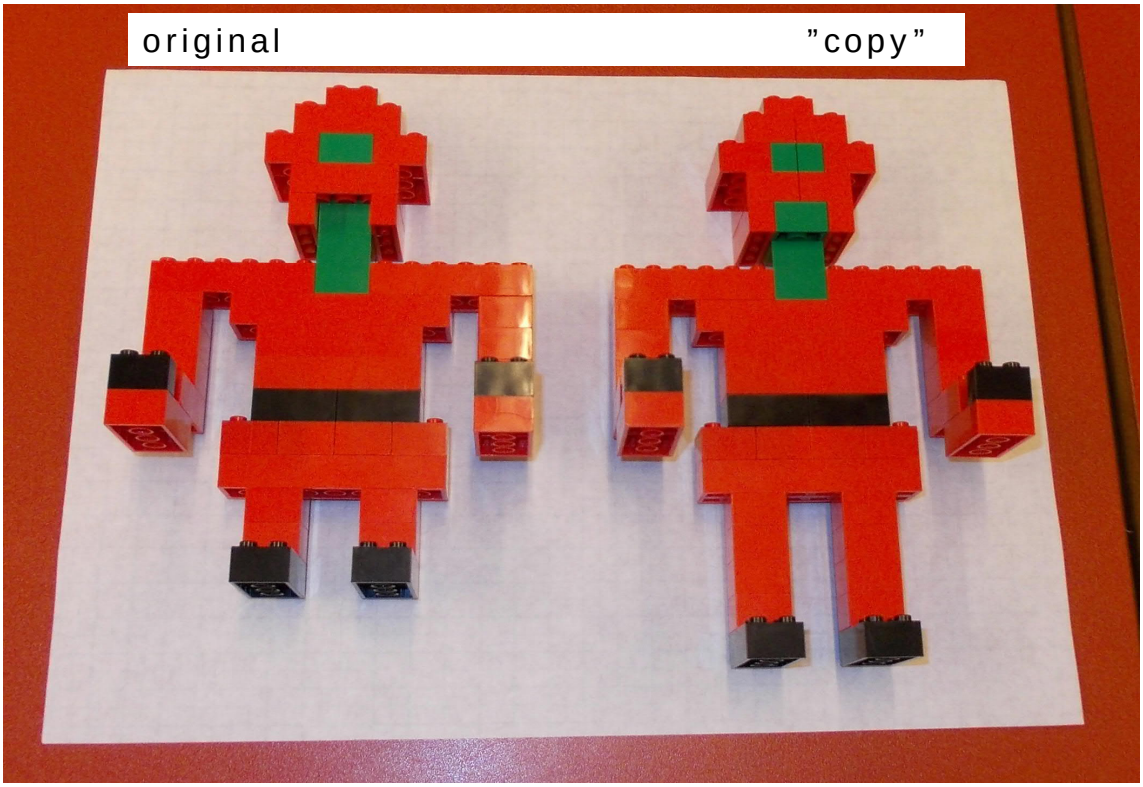
c) A new group got the specification and similar set of LEGos, to build a similar gadget.

Well well… WHY the requirements specification is said to be difficult…??

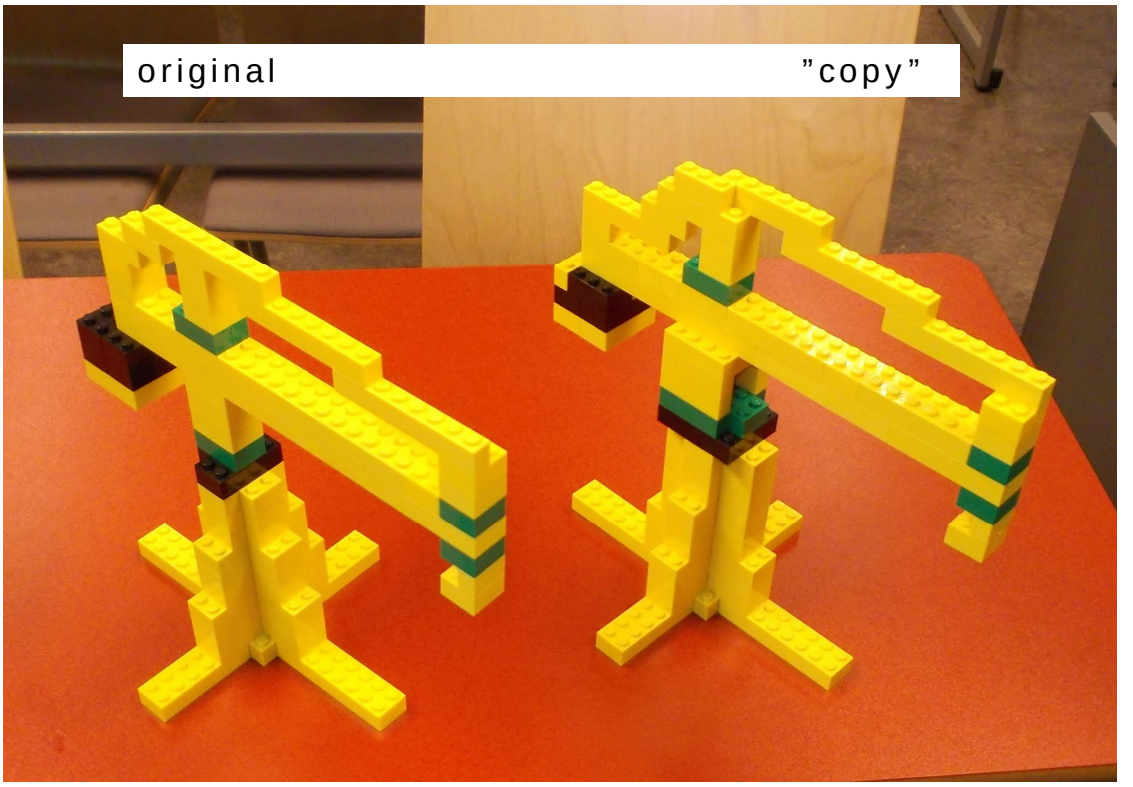
Old WE1 work.



Old WE1 work.

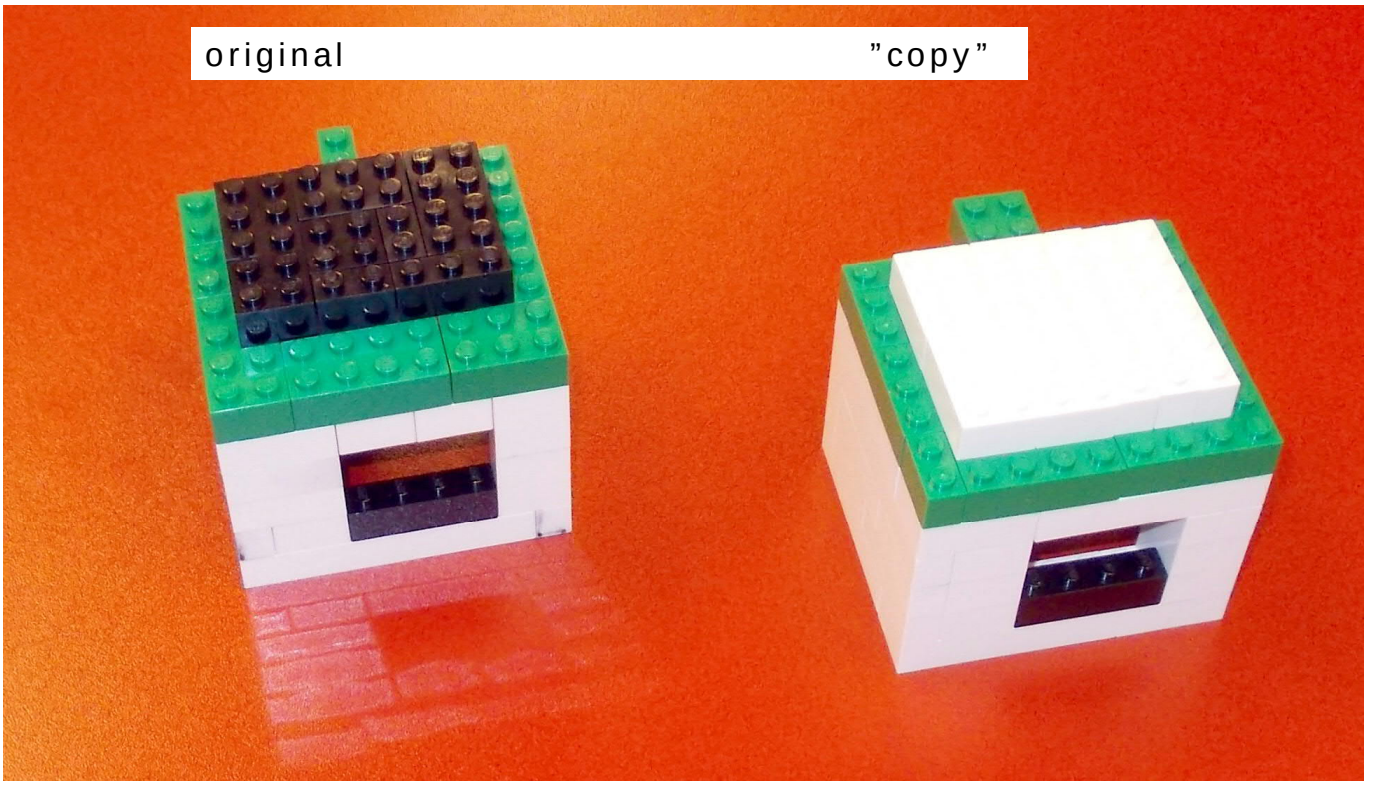


Old WE1 work.

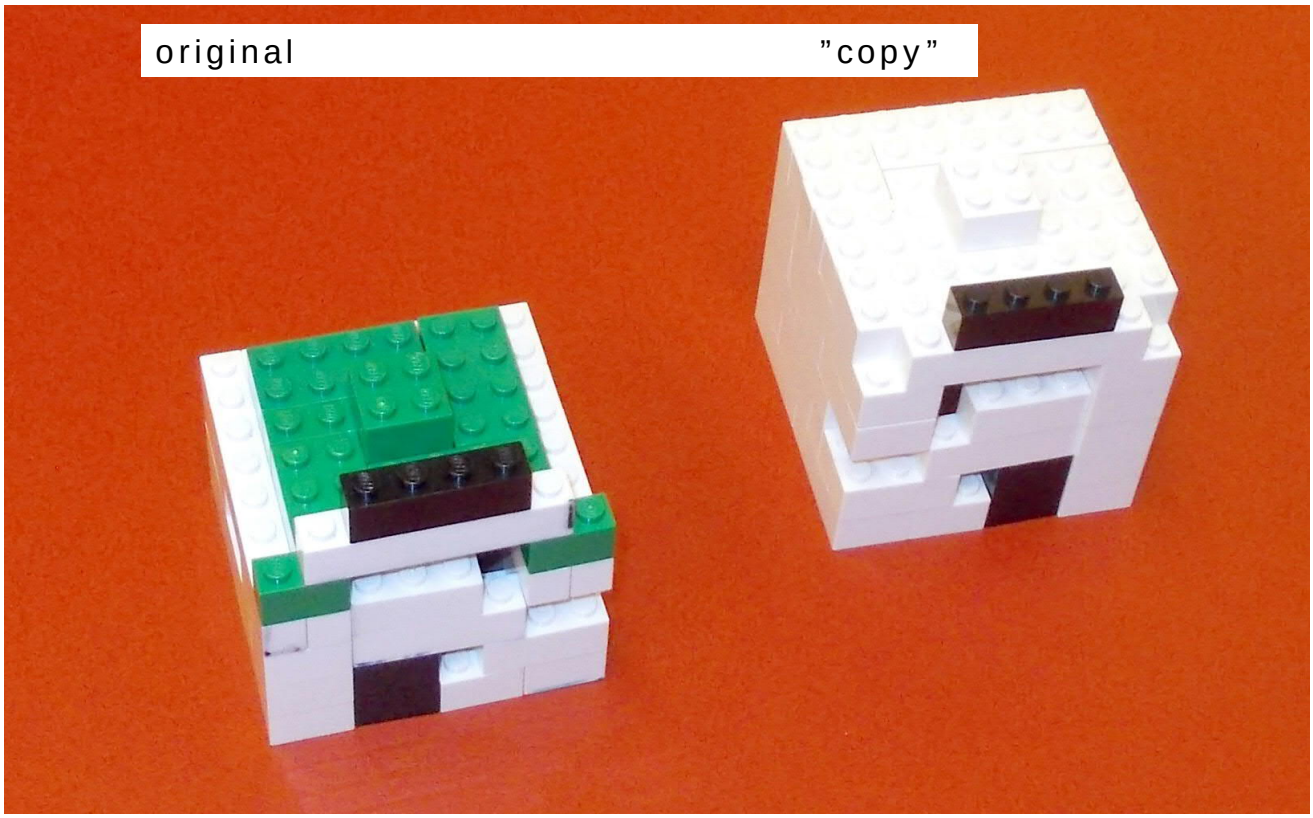


Tampereen yliopisto
Tampere University

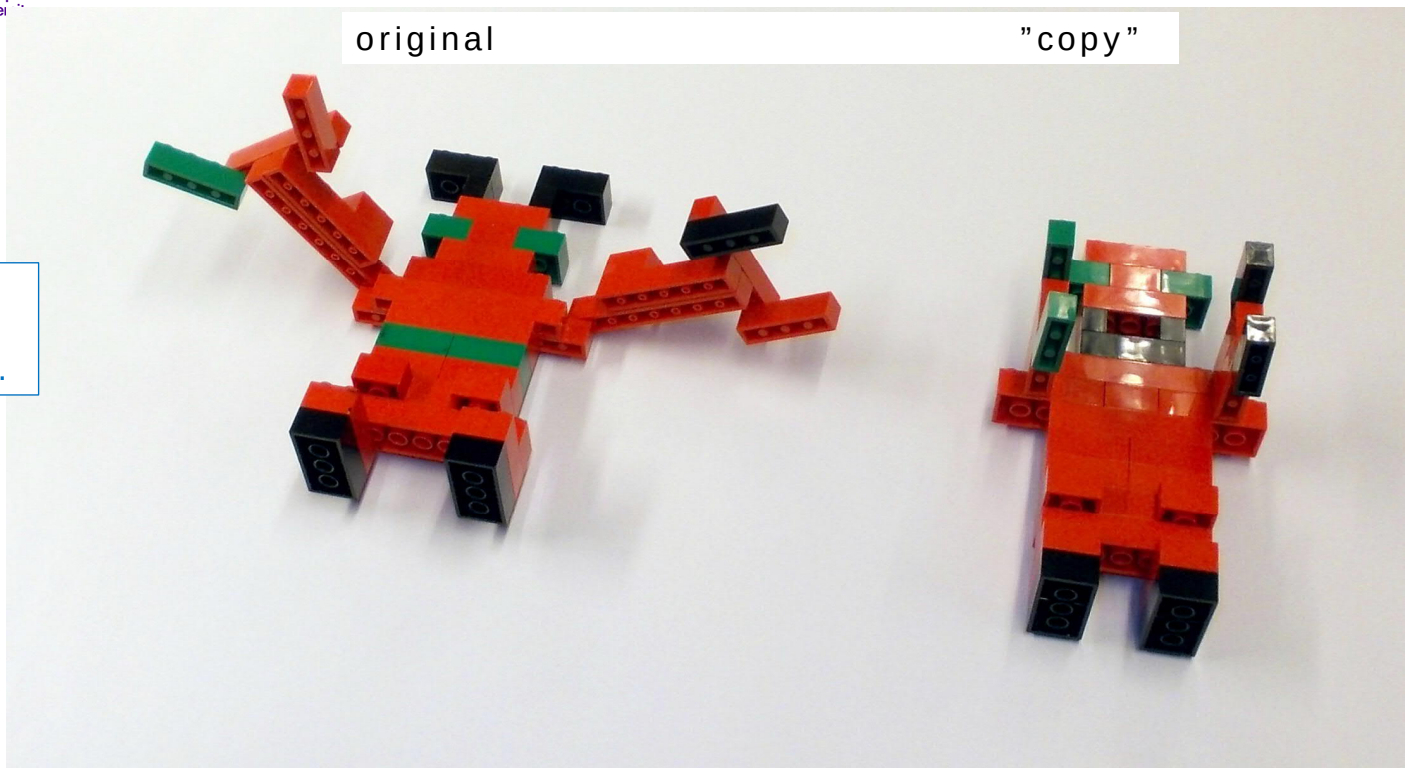
Old WE1 work.

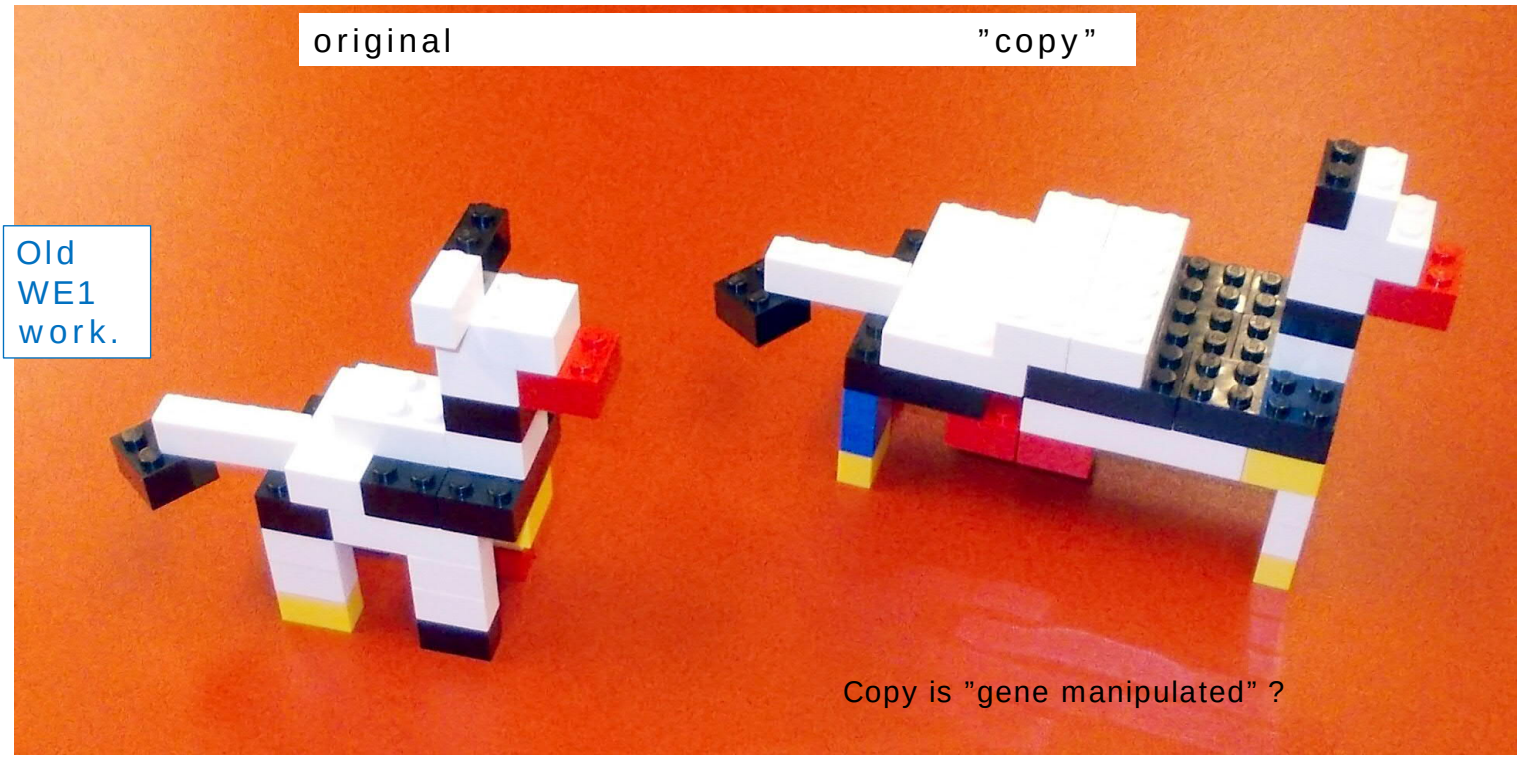


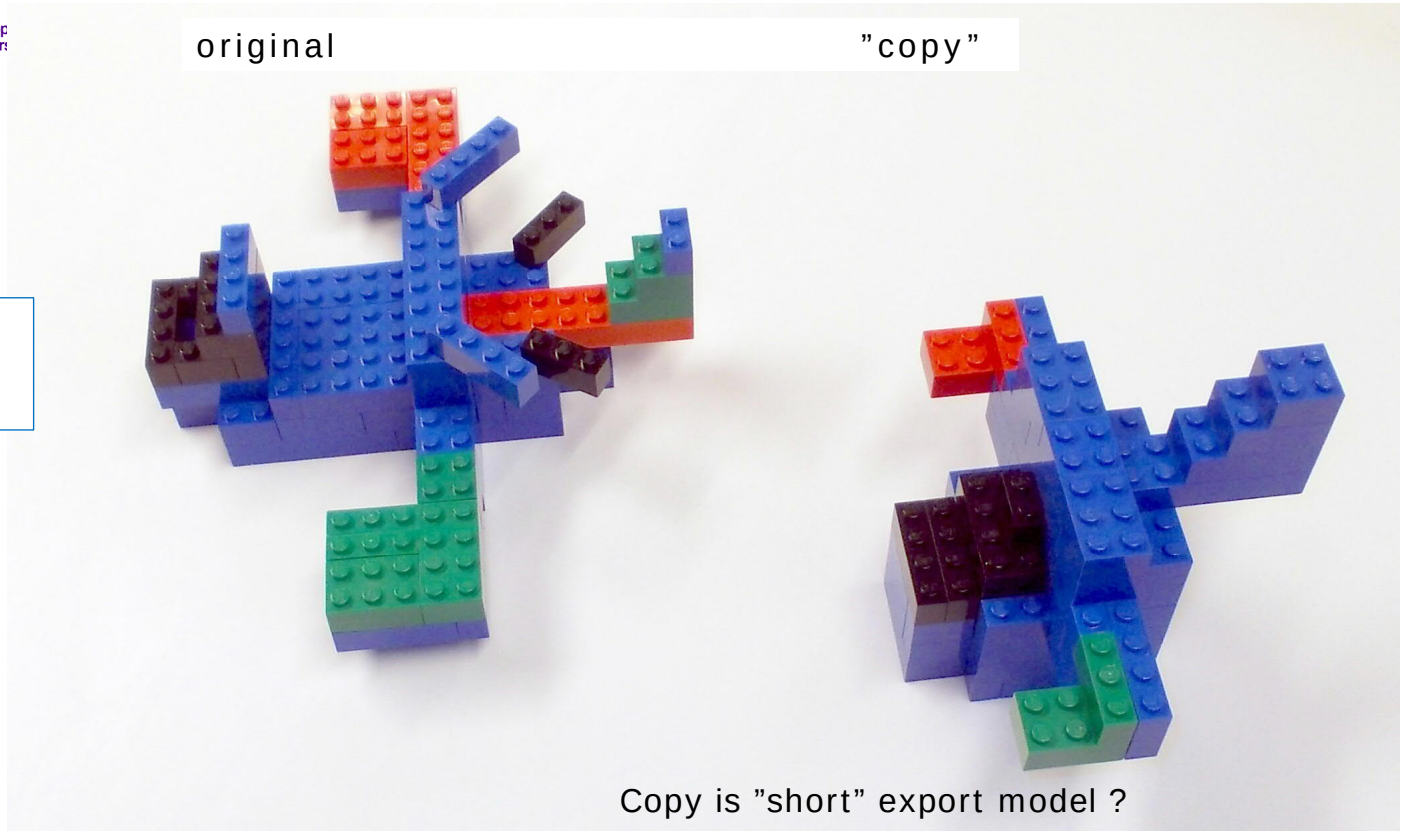
Old WE1 work.

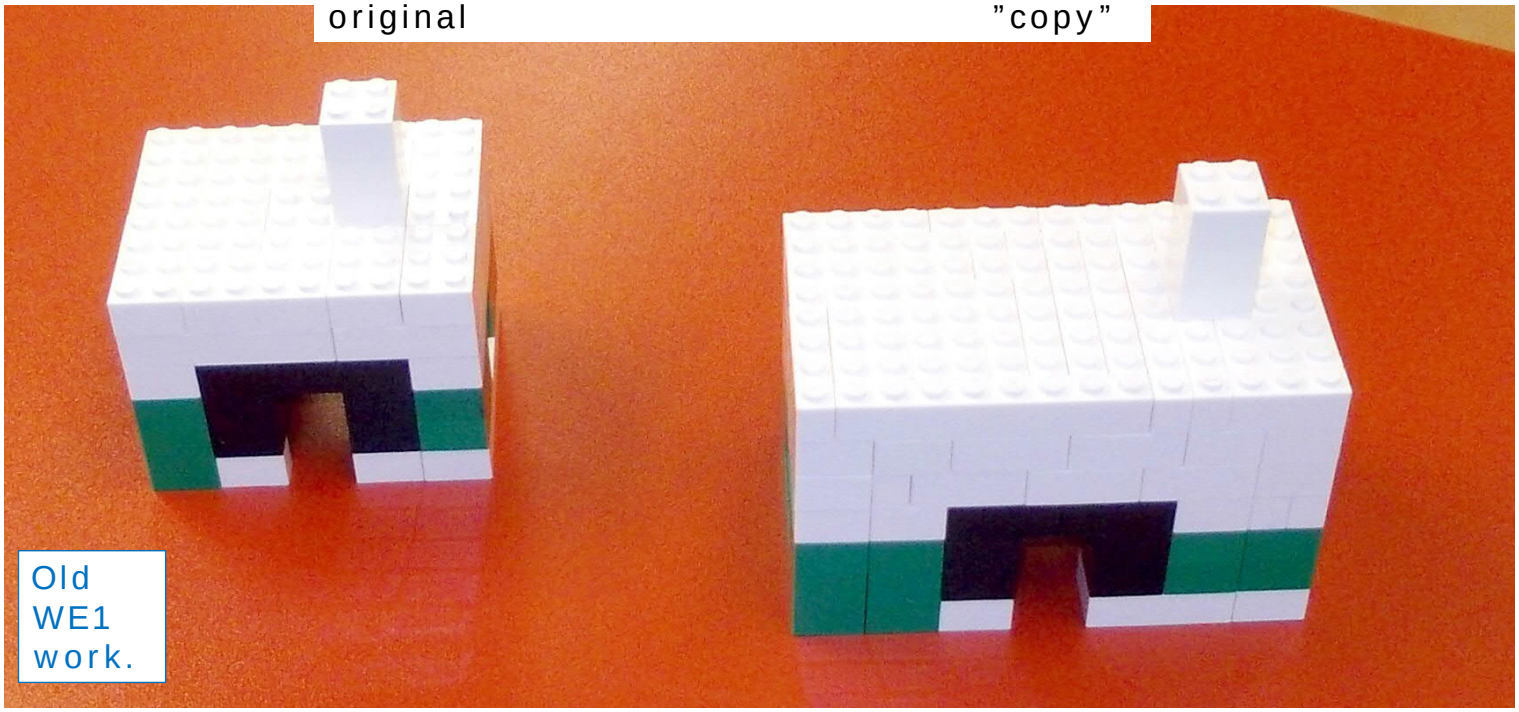


Old WE1 work.




original
”copy”
Old
WE1
work.
Copy is ”gene manipulated” ?


original
”copy”
Old
WE1
work.
Copy is ”short” export model ?



Tampereen yliopisto
Tampere University

Ariane 5 disaster, 04.06.1996

Code from successful Ariane 4 was re-used in new and more powerful Ariane 5.

e) At 36.7 seconds after H0 (approx. 30 seconds after lift-off) the computer within the back-up inertial reference system, which was working on stand-by for guidance and attitude control, became inoperative. This was caused by an internal variable related to the horizontal velocity of the launcher exceeding a limit which existed in the software of this computer.

f) Approx. 0.05 seconds later the active inertial reference system, identical to the back-up system in hardware and software, failed for the same reason. Since the back-up inertial system was already inoperative, correct guidance and attitude information could no longer be obtained and loss of the mission was inevitable.

3.2 CAUSE OF THE FAILURE

The failure of the Ariane 501 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.

The extensive reviews and tests carried out during the Ariane 5 Development Programme did not include adequate analysis and testing of the inertial reference system or of the complete flight control system, which could have detected the potential failure.

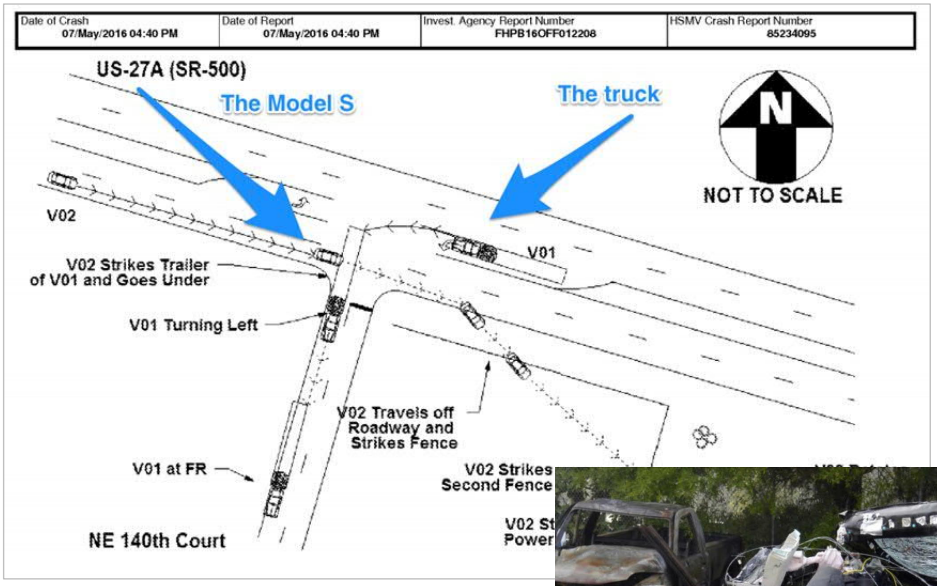
http://sunnyday.mit.edu/nasa-class/Ariane5-report.html

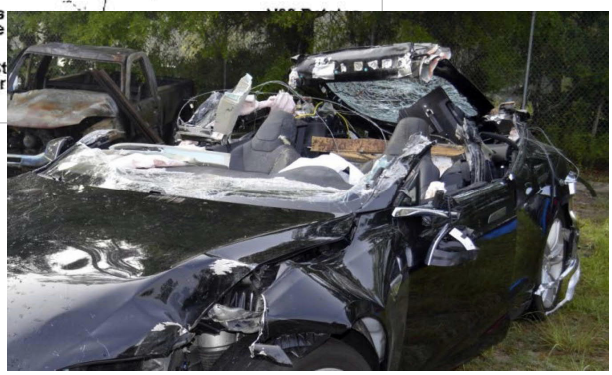
Tesla car accident 07.05.2016

- The first known death caused by a self-driving car was disclosed by Tesla Motors on Thursday.
- The 7 May accident occurred in Williston, Florida, after the driver, Joshua Brown, 40, of Ohio put his Model S into Tesla's autopilot mode, which is able to control the car during highway driving.
- Against a bright spring sky, the car's sensors system failed to distinguish a large white 18-wheel truck and trailer crossing the highway, Tesla said. The car attempted to drive full speed under the trailer, "with the bottom of the trailer impacting the windshield of the Model S", Tesla said in a blogpost.
- Eight months after a fatal crash, federal auto-safety regulators said their investigation of the car found no defects in the system that caused the accident and said Tesla's Autopilot-enabled vehicles did not need to be recalled.

Tesla car accident 07.05.2016

- A police report in the Levy County Journal said the top of the vehicle “was torn off by the force of the collision”. The truck driver was uninjured.
- In its 537-word statement on the incident, the electric vehicle company repeatedly went out of its way to shift blame for the accident. The first paragraph notes that this was Tesla’s first known autopilot death in some 130 million miles driven by its customers.
- It goes on to say that the car’s autonomous software is designed to nudge consumers to keep their hands on the wheels to make sure they’re paying attention. “Autopilot is getting better all the time, but it is not perfect and still requires the driver to remain alert,” the company said.
- "Neither Autopilot nor the driver noticed the white side of the tractor trailer against the brightly lit sky, so the brake was not applied," Tesla said.





Tesla car accident

Uber car accident 18.03.2018

- A self-driving Uber Volvo XC90 SUV killed 49-year-old Elaine Herzberg as she walked her bicycle across a street in Tempe, Arizona, Sunday night, according to the Tempe Police Department.

ARIZONA · 1 day ago

Self-driving Uber car kills Arizona pedestrian, police say

Uber's self-driving car showed no signs of slowing before fatal crash, police say

The vehicle was traveling at 40 mph

- Chief of Police Sylvia Moir told the San Francisco Chronicle on Monday that video footage taken from cameras equipped to the autonomous Volvo SUV potentially shift the blame to the victim herself, 49-year-old Elaine Herzberg, rather than the vehicle.

OPINION

Uber self-driving car accident: Who's to blame when there's no driver?

The Conversation By Raja Jurdak and Salil S. Kanhere

Posted Tue at 7:55am

Is it coder's fault…??

- "It's very clear it would have been difficult to avoid this collision in any kind of mode [autonomous or human-driven] based on how she came from the shadows right into the roadway," Moir told the paper, adding that the incident occurred roughly 100 yards from a crosswalk. "It is dangerous to cross roadways in the evening hour when well-illuminated managed crosswalks are available," she said.
- Though the vehicle was operating in autonomous mode, a driver was present in the front seat. But Moir said there appears to be little he could have done to intervene before the crash.
- "The driver said it was like a flash, the person walked out in front of them," Moir said. "His first alert to the collision was the sound of the collision."
- According to the Chronicle, the preliminary investigation found the Uber car was driving at 38 mph in a 35 mph zone and did not attempt to brake. Herzberg is said to have abruptly walked from a center median into a lane with traffic. Police believe she may have been homeless.

Now the additional L3
extra slides set ends here

Tampereen yliopisto
Tampere University

Now the additional L3
extra slides set ends here